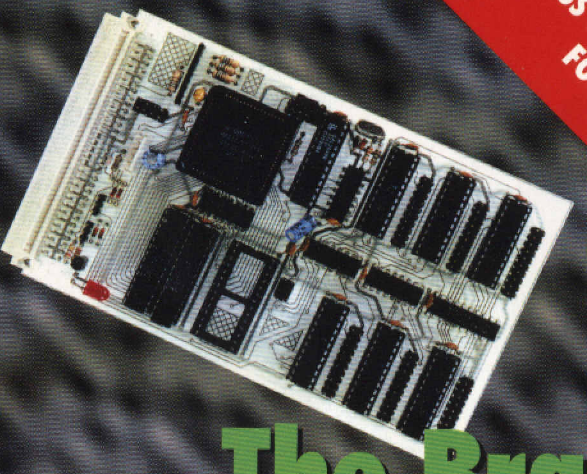
6 PHONE NOS. 44 50 58
8 64 71
30 68
37 72

ETI ELECTRONICS TODAY INTERNATIONAL

TOMORROW'S TECHNOLOGY TODAY



FREE DISK
DC-CIRCUIT SIMULATION PROGRAM
PLUS EXTRA COMPONENT LIBRARIES FOR QUICKROUTE 3 LITE

The Brains of Men and Machines

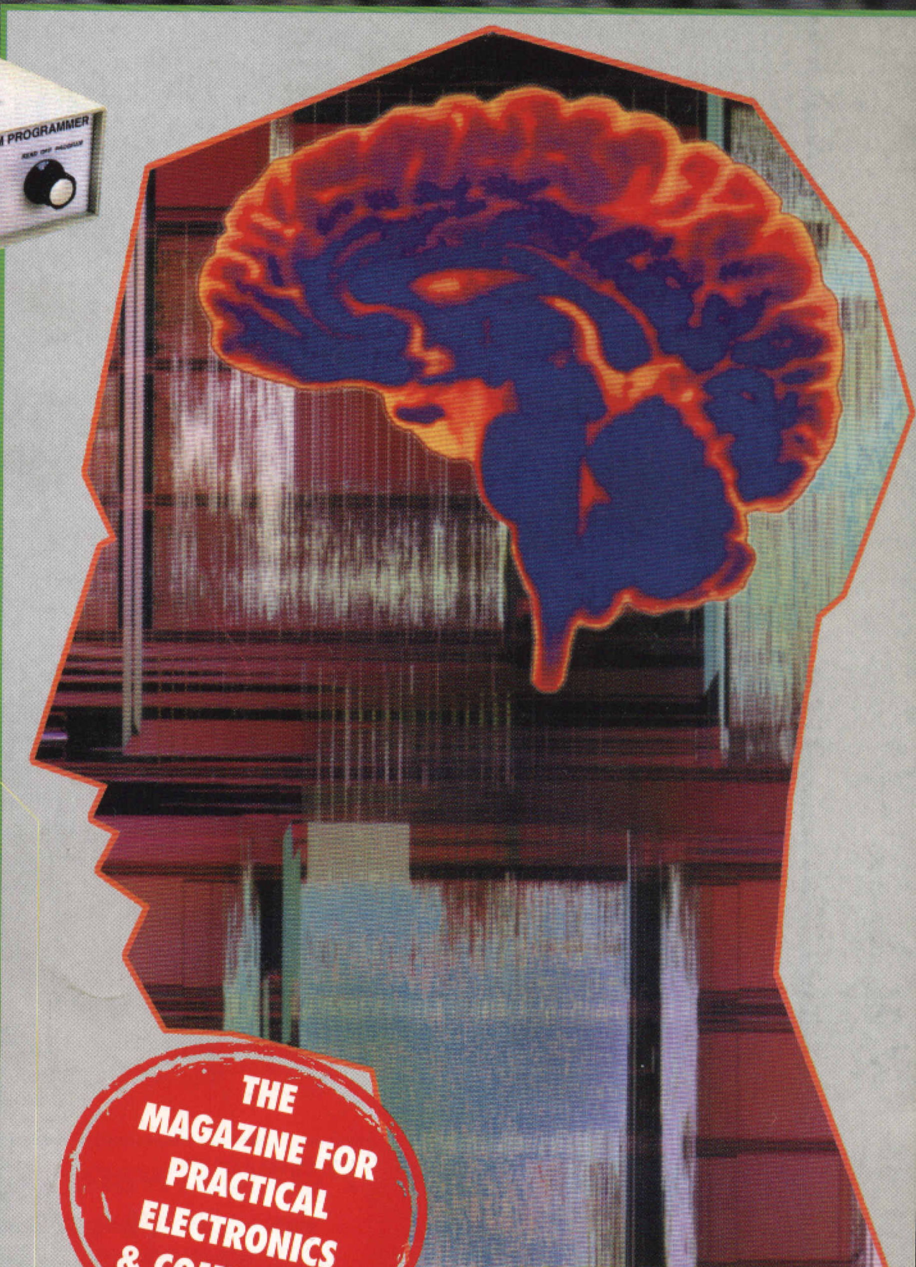
Can we build intelligent autonomous machines?

Build a low cost PC controlled EPROM programmer

Add an unlimited number of 8-BIT input/output ports to your PC

PLUS

- Testing the ETI transporter board
- Using the ETI P.O.S.T. card
- Build a simple WAH WAH pedal
- Have a cheap instrument power supply



NO COVER DISK

THEN ASK YOUR NEWSAGENT

THE MAGAZINE FOR PRACTICAL ELECTRONICS & COMPUTING

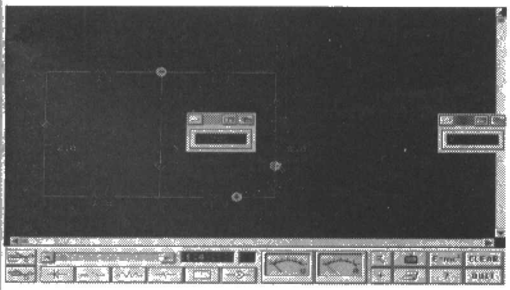
NOVEMBER 1994 £2.50

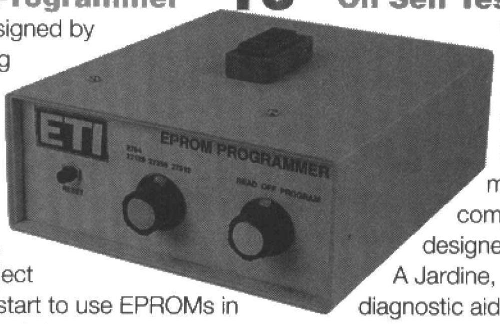
ARGUS SPECIALIST asp PUBLICATIONS

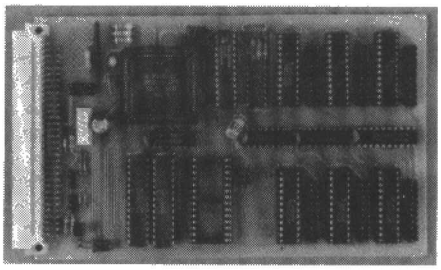
Contents

Volume 25
No.11

& Features Projects







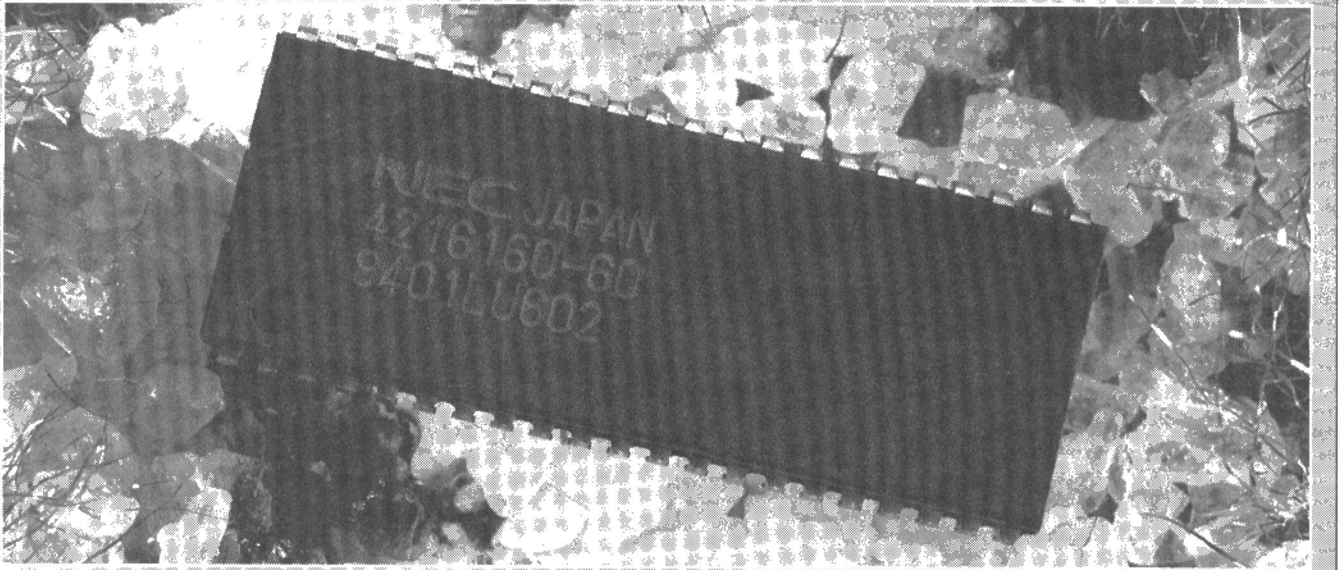
Regulars

ETI ELECTRONICS TODAY INTERNATIONAL News...

NEC first to ship 1Mx16 DRAMS

NEC Electronics has become the first manufacturer to have commenced volume shipments of second generation 16M DRAMs with a 1Mx16 organisation. The first generation devices were only available in 500mil packages which made them unacceptable to the computer industry, these devices are supplied in 300mil packages as well as 400mil 42pin SOJ, or 50pin TSOP. he uPD4216160 and uPD4218160 DRAMS are manufactured using NEC's proprietary second generation 0.45micron process, already used for volume shipment of 4Mx4 product in 300mil packages. The 1Mx16 products are available with access times of 70ns or 80ns and in 5volt or 3.3volt versions, the latter being aimed directly at the portable computer and equipment sector where very high memory densities are required.

For further details contact NEC at Milton Keynes, on 0908 691133.

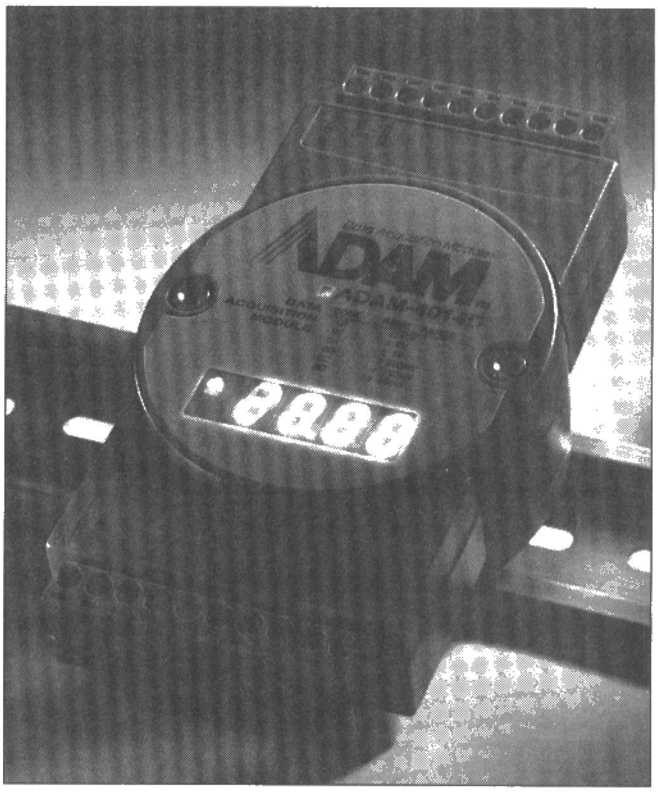


Remote measurement module with display

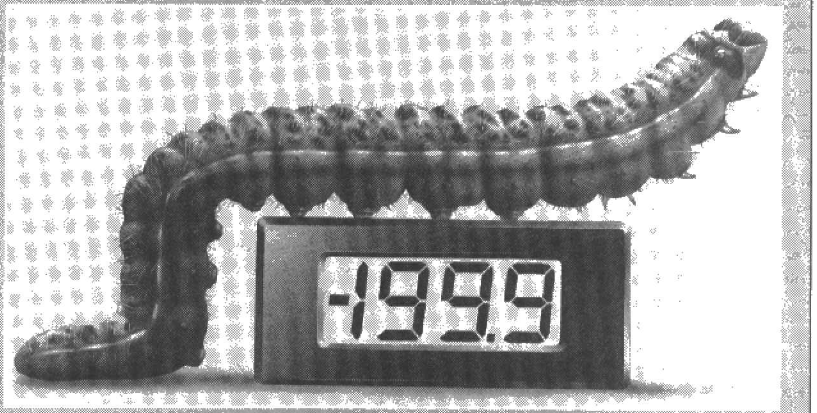
Southampton-based Integrated Measurement Systems have just launched a remote analogue measurement module with local display, the ADAM-4014D. This module provides high accuracy 16bit remote measurement of analogue or current sensors with RS485 digitised ASCII output. The module can be placed as far away as 1,200 metres from the host computer, allowing distributed remote monitoring and control which is ideal for process plants, building monitoring, outstations and many more applications.

The 4014D incorporates an on-board processor which can convert measured transducer signals to engineering units, such as temperature or pressure for example, and display the measured value on the local display, as well as transmitting measured data to a host computer via the RS485 link. The local processor can monitor measured values against pre-set high/low limits and initiate an alarm via the RS485 link when the limits are exceeded. Local LEDs on the module also indicate the alarm status condition. The 4014 module incorporates two digital outputs which can be used to trigger other equipment in the event of an alarm or as part of a local control loop.

For further information contact IMS on 0703 771143.



Ultra miniature digital panel meters.

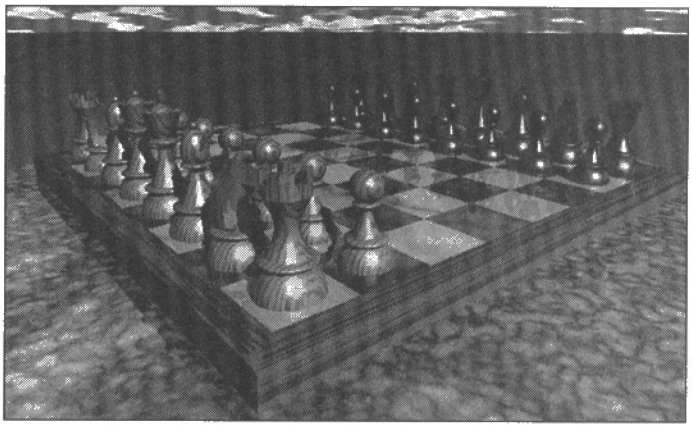


Low-cost, accurate and reliable measurements can now be incorporated into even the most compact instruments thanks to a new ultra-miniature digital panel meter from Lascar Electronics. The LCD module features 5.5mm digit height, auto polarity, auto zero, 150uA current consumption and 200mV full scale reading.

A single rail version, allowing measurements referenced to power supply ground, is also available. Both types of modules are ideally suited for integration into high volume applications, including current loop transmitters, hand-held probes, and personal gas monitors.

For further details contact Lascar Electronics of Salisbury Wiltshire, on: 0794 884567.

New low-power colour LCD display



NEC Electronics' new 9.4 inch colour LCD module has a single edge light which reduces the power consumption from the 8Watts of the previous unit to 4.8Watts. This will provide a 40% increase in battery life for the portable equipment in which this unit will be housed.

The NL6448AC30-10 is an active matrix thin film transistor (TFT) colour liquid crystal display module with a resolution of 640x480 pixels. It can display 4096 colours with a contrast ratio of 110:1 and has a 40msec response time. Although lighting is achieved by only a single edgelight, the unit has a high luminance of 90cd/m2 (typical). It has a typical horizontal viewing angle of 45 degrees and a vertical viewing angle of 30degrees.

The module consists of an LCD panel and integral LSI circuits for driving both the LCD and the backlight. The actual panel is composed of a TFT array glass substrate superimposed on a colour filter glass substrate. The liquid crystal material fills the gap between the two. Each of the liquid crystal cells acts as an optical switch which controls the light transmission from the edge light by the signal applied through the TFT switch.

For further information contact NEC Electronics UK Ltd, of Milton Keynes, on: 0908 691133

Event diary.

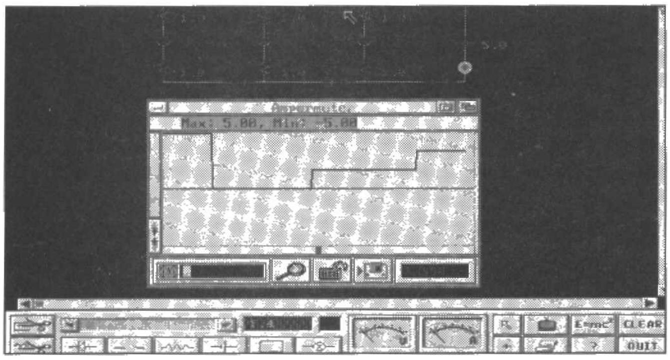
4 October	Talk on Electrical Safety and Regulations, Sudbury and District Radio Amateurs. Tel: 0787 313212
4-6 October	EDI 94, ICC Birmingham. Tel:081 742 2828
10 October	Inside your PC, a Talk by Martin Rhodes, Stratford upon Avon and District Radio Society. Tel: 0789 740073
11-13 October	Voice 94, Olympia, London. Tel: 0733 575020
24 October	QRP by Norman Field. Stratford upon Avon and District Radio society Tel: 0789 740073
26-27 October	Instrumentation, Sandown Exhibition Centre, Sandown. Tel: 0822 614671
26-28 October	PEVD 94, Power Electronics and Variable Speed Drives Conference, Institute of Electrical Engineers, London. Tel: 071 240 1871
1-3 November	Windows Epo, Olympia, London. Tel: 0256 381456
5-6 November	8th North Wales Radio and Electronics Show, Aberconwy Conference Centre & The New Theatre, Llandudno. Tel: 0745 591704
13 Nov	Midland Amateur Radio Society rally at Stockland Green Leisure Centre, Slade Road, Erdington, Birmingham. Doors open at 10am, admission £1. For further details ring 021 422 9787 or 021 443 1189(evenings only).

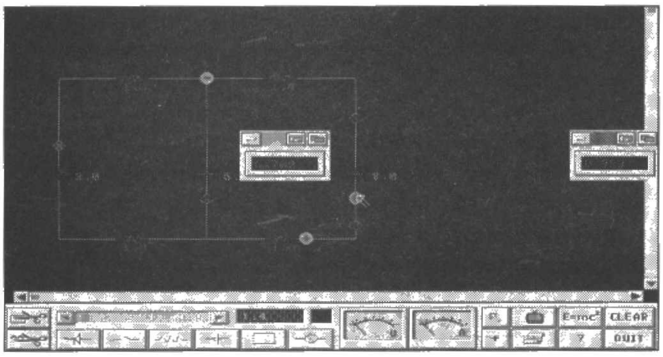
If you are organising an event which you would like to have included in this section, please send full details to: ETI, Argus House, Boundary Way, Hemel Hempstead, Herts HP2 7ST, clearly marking your envelope Event Diary.

ETI COVER DISK

Another collection of useful programs, free from ETI

This month's cover disk contains four useful programs; three are PC diagnostic and test utilities, and the other is an educational DC circuits simulation program. The disk also contains a collection of component libraries, plus the complete documentation for the Quickroute 3.0 Lite that was given away on last month's ETI cover disk.





Installing the programs

Some of the programs on the disk provided free with this issue are supplied in compressed format and thus cannot be directly run from the floppy. We have done this in order to get more onto the disk. It is therefore necessary to decompress these files before they can be used.

The following programs on the disk are stored as self-extracting archives; they are:

BENCH - 3D VGA benchmark utility

BURN - PC Burn-in test to fully exercise a system and diagnose common faults

CONV - Monitor testing utility, will check for convergence, pincushioning etc.

QR - Additional library files for Quickroute 3.0 Lite, plus the complete documentation for this program. (Note: the PCB design, and schematics layout program Quickroute 3 Lite can be found on the ETI October 1994 cover disk; copies can be obtained from: ETI Back Issues Dept, Argus House, Boundary Way, Hemel Hempstead, Herts HP2 7ST. price £2.50 plus 60p P&P.)

To decompress these programs to a hard drive we suggest the following:

Create a directory on the hard drive to hold the files

e.g. MD 3DBENCH

Change to the directory created

e.g. CD 3DBENCH

Place this disk in your floppy drive

Type the name of the self-extractor, including the path to the floppy drive

e.g. A:BENCH

The files will now be extracted to the current directory

The DC CIRCUITS program on this disk is NOT compressed and will run from the floppy. Should you wish to move the program to a hard disk, create a directory called DC and copy all files found in the DC directory on this disk to it. Next, copy the two files FONT.P{M and FONT.P{X to the route directory of the hard drive. These files MUST be in the route directory for the DC program to operate.

Using CD Circuits.

The program DC Circuits is a slightly cut-down version (file save has been disabled) of the educational program produced by Buckinghamshire-based Visual Products as part of their Visual Physics range of software.

This program is not supplied in compressed form and can therefore be run directly from the cover disk. It is a DOS program, so from the A: DOS prompt simply enter the directory called DC and run the program also called DC. A couple of pages of information about the program and Visual Products are then displayed followed by the main simulation screen.

The workspace area can be larger than the actual screen dimensions, and there are scroll bars to the right and along the bottom of the work area window which allow one to move the window around the workspace. Along the bottom is a control bar with a number of buttons and sliders (the graphics user interface employed sticks quite closely to standard Windows conventions and is therefore very easy to use).

The best way to learn how to use this program is to run the demo and the tutorial. These can be accessed by using the mouse pointer and the left mouse button to point and click on the button to the bottom right of the control bar which has an icon representation of a question mark. The self-running demo takes one through the steps of building a DC circuit, using batteries of different voltage and polarity, adding resistors, diodes and switches, editing the circuit and, of course, measuring current and voltage.

Current and voltage measurements are performed by probes from virtual voltmeters and ammeters. There can be as many of these probes as necessary and their display is shown in real time, so altering a component value will immediately be reflected in the displayed measurements. Current probes show the flow direction as well as the value, and both types of probe can show either a digital value or, when expanded, a waveform; this can be displayed with a variable timebase.

This is an excellent educational program, and one which most readers will probably find interesting and useful. If you want to get the full version then it is available from: Visual Products, PO Box 48, Great Missenden, Bucks, HP16 9HL. Tel: 0494 890601. Price £14.95 plus £1.65P&P.

Using the Quickroute 3.0 Lite files.

When decompressed, you will find a printable form of the Quickroute 3.0 Lite manual under the directory QR. It is stored in the file QR3.WRI and can be loaded into Windows Write and printed in the normal way. Please note, however, that to save space and make it easier to print out, the figures have not been included.

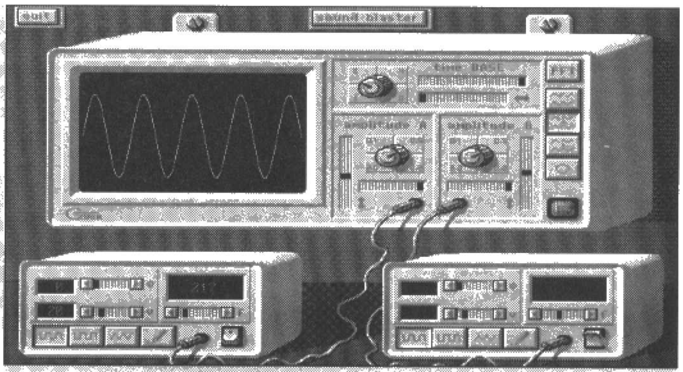
Also in the QR directory are two more Quickroute symbol libraries. The file LOGIC.SYM contains a set of basic logic gates for schematics, and PCBSMT.SYM contains a set of basic surface-mount symbols. These surface-mount symbols must be used with metric cursor measurements, and the user must select metric cursor BEFORE loading the library. The other two files in the directory are ORDER.FRM and ORDER.WRI which are text files and will print out order forms for anyone wanting to buy one of the full versions of Quickroute.

For readers who have found Quickroute 3.0 Lite in last month's ETI useful, but would like to have a more sophisticated and powerful version, the program's authors are able to offer a 40% discount on Quickroute 3.0 Designer, this will let users edit circuits that are 20 times more complex (10,000 nodes) and also includes full autorouting on one or two layers. This version normally retails for £99.00 (excluding VAT and P+P) but ETI readers who send a copy of the cover disk from this issue with their order can get it for just £58.00 (excluding VAT and P+P). It should be noted, however, that at this price a printed manual will not be included; instead, the full manual text will be available as an on-line help (where it can be printed out).

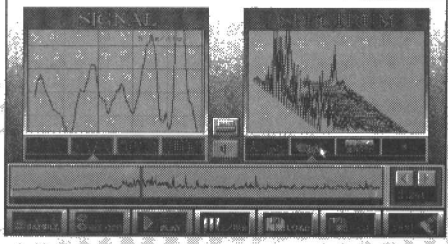
Orders should be sent to: POWERware, 14 Ley Lane, Marple Bridge, Stockport, SK6 5DD. If you have any queries about this or other versions of Quickroute then phone or fax POWERware on 061 449 7101.

The PC Utilities.

The way in which the three PC utilities are used is explained in the PC Clinic section of this issue.



Other Electronics Related Software from Visual Products



Besides their range of educational visual software products, of which DC Circuits is just one, Visual Products produce two very interesting pieces of software which will have a much wider appeal than simply for use in schools and colleges. These are a PC-based oscilloscope and a PC-based waveform generator.

With the advent of multimedia a lot of PCs are now equipped with special sound input and output hardware, the most common of which is the Sound Blaster board. This is a very sophisticated piece of hardware and in the most widely used version can handle 16-bit audio signals at frequencies of up to 44.1KHz.

These two programs make use of the fact that to perform this audio input and output the Sound Blaster is, in fact, a 16-bit analog to digital and digital to analog converter board. Both the input to and output from this board are 5V peak to peak and the output sufficiently powerful to drive a speaker.

This means that both the input and output on a Sound Blaster can easily be connected to other circuitry instead of the speaker and microphone. All that one needs to be careful to do is ensure that input levels do not exceed 5V, a simple little opto-isolator circuit might be a good solution to this particular problem

Sound Blaster Oscilloscope

Provided that you do not want to examine and analyse waveforms with a frequency greater than 44.1KHz, then this piece of software turns your Sound Blaster equipped PC into a very sophisticated laboratory instrument.

Besides displaying input waveforms it can also sample and store several seconds of input and then allow the user to examine them at their leisure. The signal can be displayed with a range of different timebases, and if required could be stored on disk or output through the speaker.

Besides sampling and displaying waveforms this program can also be used to perform spectral analysis on the sampled waveform. It can do this on either pre-recorded data or in real time. The analysis features include: Real Time 2D F/Fourier Transform, Real Time 3D F/Fourier Transform, and Real Time Spectographic Analysis. Features which are usually only found on very expensive pieces of test equipment and which should be of interest to readers interested in audio electronics, electronic music, and general experimentation.

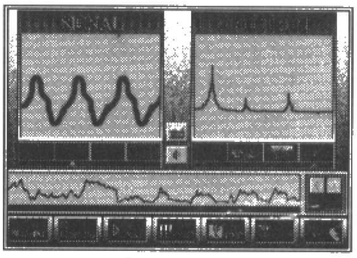
Sound Blaster Wave Generator.

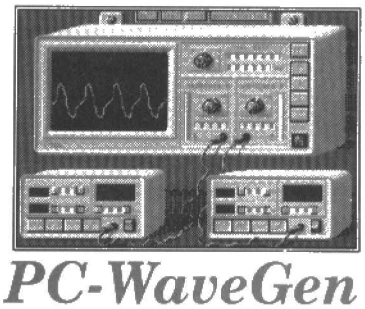
Used in conjunction with a Sound Blaster board this program will convert a PC into two easy to use wave generators, plus a virtual oscilloscope to display the chosen signals. The two waveform generators can produce the conventional sine, triangle and square waves, but in addition there is a feature to allow the user to draw his own waveform, something which should be of interest to readers interested in electronic music. The user has full control of the frequency, amplitude and phase of the output signal.

The waveforms created by each of the generators can be examined on a virtual dual trace scope, with full control of timebase and amplitude of each signal. The user can also see how the signals interact together; waveforms can be shown separated and added together. There is also a Real Time 2D F/Fourier Transform display, and a Lissajou curve display feature.

The waveforms can be output separately or together via the Sound Blaster and then fed to either the speakers or some external circuitry.

Both the above programs were originally designed for educational use, to allow teachers to show the use of what would otherwise be prohibitively expensive equipment. For this purpose they are excellent products. For similar reasons they should also appeal to a great many hobby electronics enthusiasts. The two programs are priced at £89 ex VAT each and can be obtained from: Visual Products, PO Box 48, Great Missenden, Bucks, HP16 9HL. Tel: 0494 890601.





The Brains of Men &

It may surprise many but the pursuit by scientists and researchers of ways to create non-biological systems which show some glimmerings of intelligent behaviour precedes the development of the computer, and electronics, by at least a couple of hundred years.

Thanks to the founding fathers of modern science, men like Newton, Galileo, Copernicus, and Kepler, the intelligentsia of the eighteenth and nineteenth centuries came to see the world in terms of determinate mechanical systems. They saw everything in the world as being analogous to complex clockwork systems which could be described by mathematical equations.

Just as one could build a clockwork model of the solar system which accurately replicated the movements of the planets and other celestial bodies, so they argued if one knows the equations it should be possible to build a model which replicates the behaviour of any type of system, even living ones.

It was a concept which led to the development of some truly amazing examples of mechanical automata during the early 1800s. For the most part these were mechanical birds and animals that could move and sing in almost perfect imitation of the real thing. But some automata makers were more ambitious, and even attempted to imitate intelligent behaviour by, for example, having the automata play chess.(However, this was beyond the mechanical automata makers' abilities and was invariably achieved by some form of subtle human control.)

The great engineering feats and the scientific discoveries of the nineteenth century did nothing to dispel the idea that all dynamic systems, however complex, were describable by a set of equations, and that their behaviour was therefore deterministic. Indeed they further reinforced the idea, and thanks to the work of mathematicians and philosophers it was discovered that even intellectual activities such as mathematics could be described in mechanical terms.

This was a concept which Charles Babbage, widely acknowledged as the father of the computer, attempted to implement in the middle of the last century with a mechanical computational engine. It was a task which was somewhat beyond the mechanical ability and economic resources of the time and had to wait nearly 100 years before it was achieved using electronics.

The work of people like Babbage led others, in particular the early science fiction writers, to conceive of the idea of mechanical men with mechanical brains. An idea that leapt into the popular imagination in the early 1950s with the introduction of the concept of the robot (the term robot comes from the Czech word for worker and was first coined in 1921 by the writer and dramatist Karel Capek in R.U.R). This concept was reinforced in the popular imagination by a whole host of books and B movies featuring intelligent, helpful, or malevolent metal men.

The birth of the science fiction robots coincided with two major scientific and technological developments. These were the creation of the first digital computers, and a set of major discoveries in the field of neurophysiology and psychology. The result was that people started to see the brain as a machine, a machine which could be replicated using the newly-developed computer.

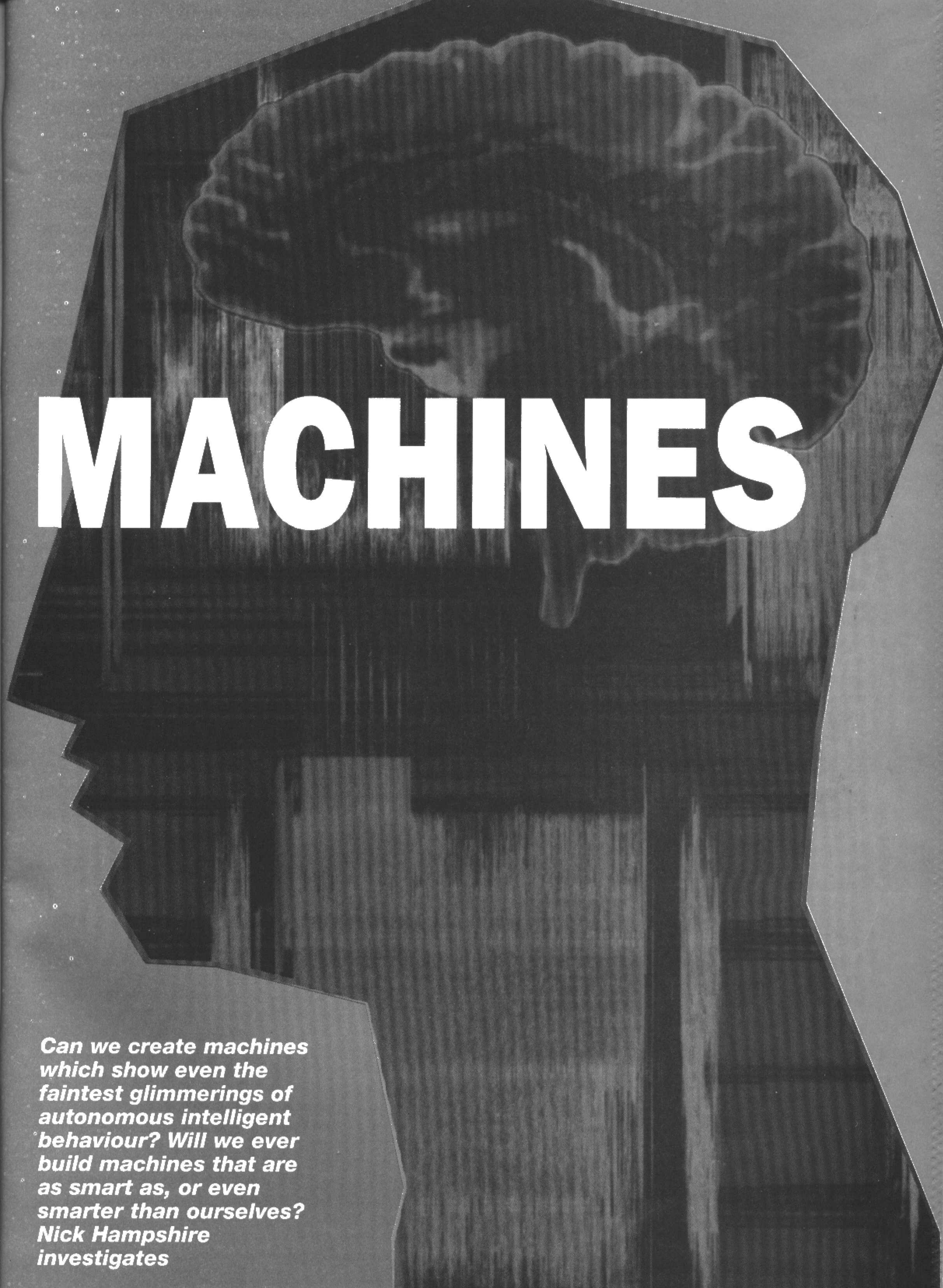
Thus in the mid 1950s we see the birth of the modern quest to build intelligent machines, a search that is usually referred to as artificial, or machine intelligence studies. These early researchers were highly optimistic. Given enough computing power they felt that they could overcome any problems, and many confidently predicted the availability of intelligent machines well before the end of this century. Indeed, according to the prognostications of these early researchers, we should have highly intelligent machines in widespread use today.

The result was that quite a lot of mainly government (in other words military) money was put into machine intelligence and allied areas such as computer language translation. The trouble was that in the early days the problems involved in creating such intelligent systems were grossly underestimated, and the capabilities of the digital computer grossly overestimated.

The truth was that neither intelligent systems or computational systems were sufficiently well understood, and the result was that optimism quickly turned to disillusionment. With failure to deliver early promises, financing dried up.

A common excuse was that computers were not big enough or powerful enough. It used to be said in the early 60s that to build a computer that would rival the human brain it would need

MACHINES



Can we create machines which show even the faintest glimmerings of autonomous intelligent behaviour? Will we ever build machines that are as smart as, or even smarter than ourselves? Nick Hampshire investigates

to be as big as the Empire State Building and consume all the power generated by Niagara Falls. It was a flimsy excuse and one that failed to convince anyone.

In consequence, machine intelligence researchers had their budgets slashed and were forced to retire to back rooms in universities - a move that encouraged them to abandon grandiose ideas and, instead, look more closely at the fundamental problems involved. The result was the development of three distinct approaches, each of which owes its origin to a different model of how intelligent systems work.

In the first approach, researchers have drawn upon the work of psychologists and developed computer software which models the various ways in which psychologists think that the brain works. The second approach has been based upon models of brain function derived from neurophysiologists and has attempted to use specialist hardware to emulate the function and behaviour of single nerve cells and groups of nerve cells. The third main approach has not come from biology or psychology but from classical cybernetics and involves the use of interlinked finite state machines to generate low-level emergent behaviour.

After over a quarter of a century of research this split still exists, but despite this each of the different approaches has had some significant successes. In the rest of this article we will look at some of these advances, and the reasons why a combination of these techniques is now seen by many as being the best way to create intelligent systems.

Machine intelligence and the digital computer

The use of conventional digital computers to model various aspects of human intelligence has probably seen the largest amount of research effort. This research has covered an extremely wide area, and includes such subjects as machine vision, natural language input and output, machine translation, knowledge systems, and robotics.

Many of the ideas produced by this research have already been applied to commercial equipment. Machine vision is used on some advanced industrial robots that are designed to perform assembly work. One can buy software that can translate text from one language to another, and systems which will allow one to communicate with a computer by means of voice commands. Natural language input is widely used to make software more user-friendly, and personal digital assistants are capable of handwriting input and analysis.

However, some of the most significant developments to come from this research are not so visible in terms of commercial products. They include the artificial intelligence computer languages such as LISP and Prolog, as well as very powerful computational concepts such as object oriented programming, parallel processing, and distributed processing.

But perhaps the most powerful of all these ideas, and one that is already the basis of quite a considerable industry is that of knowledge engineering, the so-called expert system. Such systems are now widely used to help individuals without a great deal of experience to make decisions using the knowledge of an expert on the subject; knowledge which has been stored in the expert system's database.

What is even more interesting from an electronics and computer point of view is that expert systems are now beginning to be widely used to provide larger systems with embedded intelligence. Thus a small processor may be built into a large machine tool, the function of the processor being to monitor the current state of the machine.

This would involve monitoring things such as bearing temperatures, lubricant levels, stress and strain within specific components, etc. The processor would be running a special expert system which would contain a comprehensive knowledge base about the system that draws on all the skills and knowledge of the designers. This embedded expert system is then able to detect potential faults before they occur, or help engineers locate faults when they do occur.

Of course this could be done with a conventional computer program, but expert systems have one enormous advantage; the knowledge that it uses is not contained in the code as it would be in a conventional program, but stored as a data base which can be easily maintained and upgraded without any need to rewrite the entire program.

Another advantage of an expert system, thanks to its artificial intelligence origins, is that it can be made to deal with probabilities rather than certainties. This feature would be particularly important in the maintenancing monitor system where a set of measured variables could have a number of causes and a number of different consequences. An expert system will be able to inform the maintenance engineers that there is a probability that a fault will occur in X and another probability that it will occur in any of A, B, C, or D components. This makes it far more sensitive to minute, but otherwise significant, changes in a system's state.

Using an embedded expert system might, therefore, be the ideal solution to a particular problem in a wide range of different areas that are of interest to the electronics enthusiast. They could, for example, be used to monitor security and environmental control systems, an intelligent radio control system, or even an intelligent music synthesis system, not to mention of course the robot which mows the lawn.

Expert systems have a considerable number of practical uses and can, in a very narrow field of expertise, quite effectively model the knowledge of a human expert. But they do have one major shortcoming. This is that before an expert system can be created, the knowledge base has to be extracted from the human experts.

This process of creating a knowledge base is known as knowledge induction, and in the vast majority of cases it is done by a human knowledge engineer. But in certain areas of expertise where there are only a few clear cut rules it is possible to create programs which will automatically extract a set of rules from an assorted collection of data that relates to the subject.

The problem with automatic rule induction systems is that they can function in a very arbitrary manner and create knowledge bases which are rather convoluted and obscure. By and large, this is something that workers in the field of artificial intelligence hate, they prefer systems that can be clearly understood and verified by a human researcher.

Although there have been numerous attempts at building expert systems which can create their own knowledge bases, few have been very successful when applied to real world applications. One of the handful of systems for which this statement is perhaps not true is the SOAR system developed by a group led by Allen Newell at Carnegie Mellon University in the USA.

The SOAR system (SOAR apparently stands for State Operator and Result), when confronted with any problem, will first check its own knowledge base for potential solutions. If no solutions are found then it sets up what is referred to as a problem space. It then uses special techniques to try and find all possible solutions within that problem space. Once a solution has been found it is compiled into a rule which will then be activated next time that problem is encountered.

The trouble with experimental systems like this is that they have a tendency to fall over very quickly. The reasons for such failures are manifold. In some it is probably due to errors in the induction system, in the rules which the system is 'born' with. In others it is probably due to the knowledge base becoming unwieldy. Thus a system with a hundred rules may work perfectly but one with ten thousand rules may well slow down to a speed at which it effectively ceases to work.

This is a major problem facing designers of machine intelligence systems using this type of technique. Already it has been calculated by researchers on the Cyc project (this is a project backed by MMC, a consortium of 56 US high technology computer companies which has as its goal the construction of a 'common sense' knowledge base) that about 10 million facts are required before a system will be able to pick up knowledge for itself more easily by reading than by being given the facts by a human researcher.

So far after nearly five hundred programming man-years of effort over a ten-year period, the Cyc team have given their system about 1.5 million facts. There is thus a very long way still to go before they can find out whether their calculations are correct and the system starts to learn facts for itself.

Special computational hardware and machine intelligence

Some researchers in artificial intelligence have for a long time rejected the psychology derived computer program models of intelligent behaviour and instead opted for models based upon the work of neurophysiologists. For these researchers the goal of artificial intelligence will only be reached by the use of special computational hardware, such as the now widely-used neural network.

The development of the neural network concept is one that has a long history, indeed the widely acknowledged father of neural networks, Bernard Widrow of Stanford University, did his initial work in this area in the late 1950s. This work culminated in his 1963 neural network weather forecasting system, a system that was correct about 83% of the time (the local human forecaster at the time was only 65% correct).

This early work on what is known as connectionism, was then carried further forward by people like Marvin Minsky and Seymore Papert at MIT with their development of the percepteron.

Unfortunately, their exhaustive mathematical analysis of percepterons published in 1969 almost put an end to work in this area and the concept languished in the academic doldrums for nearly 15 years.

It was then realised that although Minsky and Papert's analysis was correct it actually only applied to a particular type of neural network; ones with a single layer. This realisation has led to an explosion of interest in neural networks and their application to a solving a wide range of different problems ranging from pattern recognition to process control.

So what exactly is a neural network? More correctly, we should ask what is an artificial neural network since our own brain is, in fact ,a neural network. In other words, a network of neurons or nerve cells. The definition of a neural network starts therefore with the definition of an artificial nerve cell.

The artificial nerve cell basically consists of a summation unit with a large number of inputs and a single output. Each of the inputs has an excitatory (positive) or inhibitory (negative) weighting. These weights are created by a feedback loop that is part of the nerve cell's learning process.

An input line when stimulated will thus produce either a negative or positive signal, and the sum of these signals will determine whether the artificial neuron 'fires' and produces an output. If the output is correct, then the weighting that produced that output in response to the given pattern of input stimulation is re-enforced; if the output is incorrect then the weighting for that pattern is weakened. In this way, a nerve cell will learn to recognise input patterns and classify them into those which cause it to 'fire' and those which do not.

An artificial neural network consists of a number of these artificial neurons connected together in a layered structure. The first layer will consist of a number of neurons, the inputs to which are connected, perhaps, to some form of sensor array. The result of applying this first layer to the sensor array is a pattern of outputs from each cell, a pattern which varies as the pattern of sensor inputs varies (note that this is a system very similar to the early percepterons that were discredited by Minsky and Papert).

The output from each of the neurons in this first layer is then connected to the inputs of each of the neurons in the second layer. These in turn are connected typically to a third layer which produces the actual classification results. Such a three-layer network is known as a hierarchical network and is the most powerful form of neural network, and the type employed in complex applications such as speech recognition. This is because the second layer, the so called 'hidden units', can generate their own internal representation.

The actual computation within the network thus consists of a process of spreading activation. After the initial weights are set, entering data into the system causes it to pass through a series of state changes that ultimately end in stability. These state changes equate with changing weight values, stability is when weights cease to change.

The weighting associated with each input typically ranges from +1 to -1 and training a neural network is a matter of adjusting these weights. The training can be in either supervised, unsupervised, or self-supervised mode. Supervised learning occurs when the user provides it with trial and error inputs and then teaches it correct or incorrect responses. In unsupervised systems, data is entered without human intervention leading to the creation of 'data clustering' within the network. Finally, self-supervised learning takes place when the system itself monitors itself and corrects any errors by feedback through the network.

It is this learning capability which is one of the strongest features of neural network systems. In particular when coupled with the technique's outstanding pattern classification capability and its tolerance to faults in individual neurons. Indeed neural networks can be used to classify patterns which the user cannot detect; a typical example is the neural nose system looked in ETI a couple of months ago. Here the inputs from the sensors might vary from system to system, but it would always produce the correct output, even if some of the sensors ceased working.

The majority of neural networks are, in fact, not built using special hardware but simulated on a conventional digital computer. It is easier to research the subject using this approach since a computer model is easier to modify than a physical one, and in commercial applications it is often much cheaper to use a computer software simulation. However, neural network chips are now available from a number of different sources and it is thus possible to construct a genuine hardware-based neural network system. Indeed, some forecasters are predicting that neural networks will account for a very substantial chunk of the electronics/computing/robotics market within the next decade.

Today there are already many commercial and military products which use neural networks. Typical examples are the systems used at airports to detect explosives and drugs, systems which have proved very successful even when faced with cleverly-concealed substances or hard to detect plastic explosive. Neural networks are also being used in control systems of modern jet fighters, and of course in speech and character recognition systems.

But what about the use of neural networks in the creation of machine intelligence? Here the question mark is very large. The human brain has somewhere in the region of 10 billion neurons; very few practical neural network systems have been created with even a 100,000 neurons, let alone a few million. Neural networks have proved to be excellent at pattern classification, but by all accounts have yet to demonstrate any rudimentary forms of intelligent behaviour. It is not impossible that they will, but it will require a lot more work and, in the final analysis, our own brain is the only model neural network researchers have of a functioning intelligent system.

Emergent behaviour as the route to machine intelligence

The conventional approach to designing machines which display some form of intelligent behaviour is an analytical one. The analysis looks at the kind of input the system will receive, and how it will have to respond to them. It is on the basis of this analysis that the system is designed. But there is of course one fundamental flaw in this design approach. What happens if the system encounters input which was not part of the initial analysis.

Instead of this 'top down' approach, the alternative is a sort of evolutionary approach which starts with a very simple basic system that is gradually allowed to grow and develop its own behavioural patterns. This is a technique which is being developed by a team under Rodney Brooks at MIT, an approach which he calls subsumption and which has led to the creation of a whole series of small, insect-like autonomous robots or insectoids.

A good example of what Brooks' team mean by subsumption is a foot-long robot called Ghenghis. It is a robot that can walk around rough or sloping terrain and which can avoid obstacles. Indeed, it has even been trained to pounce on passing humans. It is a robot which shows comparable behaviour patterns to insects, but it is a robot which was not designed by the analysis route, in other words its behaviour was not pre-planned but emerged in a gradual manner.

What dictates Ghenghis' behaviour is a collection of interrelated and communicating processes, each of which is in fact an augmented finite state machine, or AFSM. Each of these AFSMs is a tight loop coupling sensors and actuators, and controls a specific aspect of behaviour, and contains numerical information which defines its current state. This current state can be changed by input from environmental sensors or from other AFSMs. The subsumption process implies starting off with a system that has just the simplest behaviour and then adding AFSMs to enrich that behaviour.

This is exactly how Genghis and his fellow MIT insectoids developed. The basic level of behavioural system is the leg, of which Genghis has six. Each leg consists of two motors and a set of foot sensors, and its behaviour is controlled by a communicating system of five AFSMs. These work as a closely linked system that is duplicated for each of the six legs.

Indeed, the distribution of control goes as far as having a separate microcontroller chip for each leg. The five AFSMs that run in each of these processors spend all their time checking the position of the leg and keeping it standing in the proper attitude and with the appropriate pressure applied to the ground.

The important thing to note here is that each leg is an independent system of linked AFSMs. By themselves the legs will be unable to perform any co-ordinated walking action, they will just wave around in the air and react solely to pressure, or absence of pressure, on the foot sensors. Actual co-ordination of leg movement for walking is controlled by the next level of subsumption, by two other AFSMs, the Balance AFSM and the Walk AFSM. Between them these two AFSMs ensure that Genghis will walk with a steady tripod gait and keep a steady balance, even on rough ground.

All this should have a ring of familiarity for readers who are familiar with cybernetics and control theory. It emphasises the fact that the behaviour of the system grows from the behaviour of its parts. In other words, complex behaviour patterns are just a collection of simple behaviour patterns. There is no use of sophisticated reasoning to initiate its behaviour and at a higher level it relies solely on the nature of the outside world to guide its actions rather than some internal model.

It is reacting to the world with simple rules that are embodied in its AFSMs as a result of these AFSMs incorporating simple learning capability. Why should the robot necessarily plot the position of an obstacle when it has a sensory system that can detect that same obstacle when it meets it again?

At MIT they have shown that subsumption systems can generate fairly complex activity in an otherwise fairly simple system, and researchers are confident that they can push this level of complexity, certainly as far as being able to incorporate beliefs and motivations into a subsumption robot. This could indicate that subsumption architecture might be the ideal basis on which to build other more complex systems. Once again time will tell.

Into the future.

In this article we looked at the three different approaches to machine intelligence which are most commonly used today, software which runs on a digital computer, special hardware which mimics the behaviour of nerve cells and special hardware systems based on cybernetic principles. As we have seen, all three approaches have undoubted benefits but they also have serious shortcomings. So where does this leave us?

The truth is that probably all approaches are correct. When we finally come to building an intelligent machine it will probably incorporate a great many of the different techniques which have already been developed, plus others which have yet to be invented. There is probably no universal algorithm for machine intelligence, neither is there necessarily only one way to achieve it.

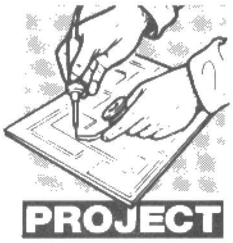
It will probably be well into the next century before we do manage to create a truly intelligent machine. It is not something we will be able to do in one giant leap, even if we poured a lot of resources into the technology, it is something which will happen gradually as, bit by bit, the jigsaw puzzle falls into place.

In the meantime, we can expect to see more and more systems which incorporate certain elements of intelligence. Commercial products with embedded neural networks and expert systems are already on the market, and in the next few years we will see subsumption-based robots exploring other planets in the universe and inhospitable parts of our own planet.

The intelligent machines are coming. The question is when?

Postscript: - The editor of ETI, Nick Hampshire, spent several years in the early part of his career working in the field of artificial intelligence. He firmly believes that machine intelligence is possible, and will be achieved one day in the not too distant future. He is also a firm believer that it is a subject in which the skilled and informed amateur in both electronics and computing can carry out significant original research work.

In fact there is little reason to believe that the heavily-funded research lab has any particular advantage in this area over the lone individual working in a back room. Both are equally likely to come up with key ideas which could eventually lead to the construction of intelligent machines. For this reason ETI will be covering some aspects of machine intelligence in greater depths, both theoretical and practical, over the forthcoming issues. These articles will include a look at neural networks, at expert systems and at subsumption systems for use in robotics.



EPROM PROGRAMMER

Commercial EPROM programmers can be very expensive. Paul Stenning shows how to build an economy version that can be controlled by your PC.

A common requirement for the electronics hobbyist today is a cheap straight-forward EPROM Programmer, suitable for occasional use, and without the extra features found on many commercial units. The EPROM Programmer presented here is designed to address this requirement, and can be used to program the standard 27 family of devices, from 2764 to 27512. It can be used with any IBM PC compatible computer as the controller and data source.

The 27 series family of EPROMS has now been around for many years, so long in fact that the smaller 24-pin devices in this family (2708, 2716 and 2732) are almost obsolete and are now more expensive than a 2764. For this reason it was felt to be unnecessary to accommodate these devices in this design since they would significantly increase the complexity.

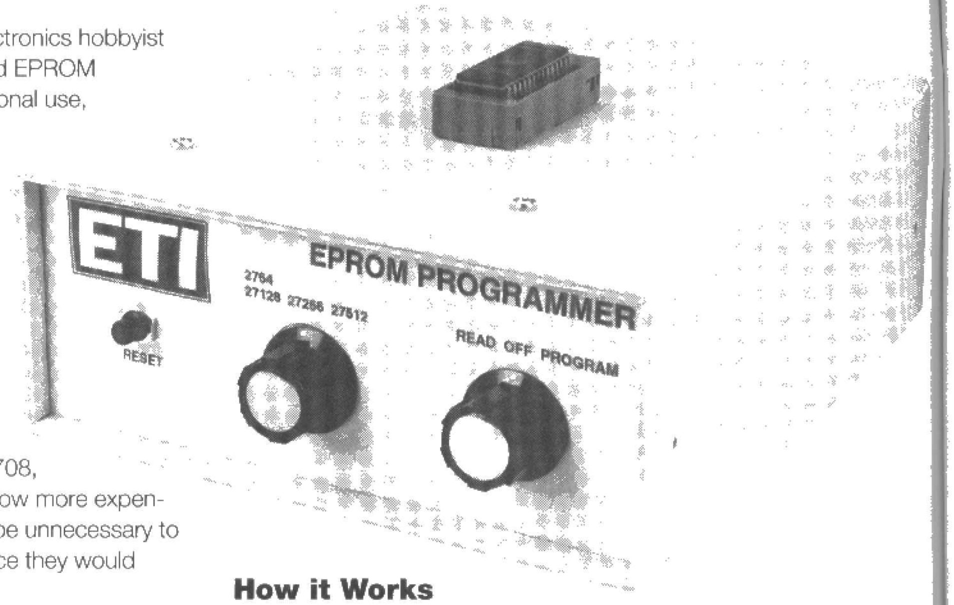
The design presented here has avoided the usual "Catch 22" situation of requiring a programmed EPROM to make the EPROM programmer work, and the design uses readily available components to reduce the likelihood of obsolescence. The unit is powered from an external PSU, supplying +5V @ 200mA and +12.6V @ 200mA. A suitable power supply is featured elsewhere in this magazine.

The programmer itself is dumb and is controlled by the host computer via the RS232 serial port (COM1 or COM2). Device selection and operation mode is set by front panel switches.

The accompanying software is available on disk from the author. The software will operate on any PC running MS-DOS or PC-DOS version 3.0 or later and having at least 512K of RAM and one RS232 serial port. A hard disk and a colour monitor are strongly recommended. The software is written in Microsoft QuickBASIC V4.5, and the full source code is supplied for those wishing to enhance or modify it. This source code is also compatible with QBASIC, as supplied with MS-DOS 5.0 and later. You do not need QuickBASIC or QBASIC to use this disk. The disk also contains the software for a matching EPROM Emulator planned for publication next month. A description of the software operation is given later.

The unit may also be suitable for use with other types of home computer having an RS232 serial port, although this has not been tested and no software is available. It will definitely not work with Commodore Amiga computers, due to a peculiarity in the serial port handling.

Please note that the programming algorithm used may not be exactly as specified in some EPROM manufacturers' data sheets. Because of this, the unit cannot be guaranteed to program every device successfully. However, no problems have been experienced to date.



How it Works

The circuit diagram is spread over a number of illustrations. Although it may initially look complex, it is in fact relatively straight-forward. When a "-" follows a signal name (for example STROBE-), this shows that the line is active low. When a number is followed by an "h" this indicates that the number is hexadecimal.

The RS232 (serial) interface, buffering and clock are shown in Figure1. IC3 (6402) is a UART (Universal Asynchronous Receiver/Transmitter) which basically converts serial data to parallel and vice-versa. The device supports most common serial data formats; in this application it is configured to give eight data bits, one stop bit and no parity checking. The data rate is set by the crystal controlled clock circuit (IC2), which in most cases will be set to 9600 Baud.

IC1 is the RS232 buffer, converting the 5V data from the UART to the +/- 10V RS232 standard, and vice-versa. This device contains a voltage doubler and voltage invertor circuit, producing +/- 10V rails from a single +5V input. The four external capacitors (C1 to C4) are required for these circuits.

A byte of serial data arriving on pin 20 of IC3 will be converted to parallel data, which will appear on pins 5 to 12. Pin 19 will go high, and no further data can be received until pin 18 is taken low momentarily. One gate of IC4 causes this to happen, R3 and C8 create a slight delay giving the STROBE- pulse adequate width for clocking other devices. When pin 23 is pulsed low, the parallel data on pins 26 to 33 is transmitted in serial form from pin 25.

R4, D1, C9 and one gate of IC4 produce the power-on reset pulse for IC3, IC5 and IC6. SW1 allows the circuit to be reset again as required.

The pinouts of the four EPROM types handled by this unit are shown in the table below.

It can be seen from the above that most of the pins are the same for all devices. Only 5 pins require special attention; these are 1, 20,

PIN	2764	27128	27256	27512
1*	VPP	VPP	VPP	A15
2	A12	A12	A12	A12
3	A7	A7	A7	A7
4	A6	A6	A6	A6
5	A5	A5	A5	A5
6	A4	A4	A4	A4
7	A3	A3	A3	A3
8	A2	A2	A2	A2
9	A1	A1	A1	A1
10	A0	A0	A0	A0
11	D0	D0	D0	D0
12	D1	D1	D1	D1
13	D2	D2	D2	D2
14	GND	GND	GND	GND
15	D3	D3	D3	D3
16	D4	D4	D4	D4
17	D5	D5	D5	D5
18	D6	D6	D6	D6
19	D7	D7	D7	D7
20*	CE	CE	CE/PP	CE/PP
21	A10	A10	A10	A10
22*	OE	OE	OE	OE/VPP
23	A11	A11	A11	A11
24	A9	A9	A9	A9
25	A8	A8	A8	A8
26*	NC	A13	A13	A13
27*	PP	PP	A14	A14
28	VCC	VCC	VCC	VCC

NC	=	No Connection
PP	=	Program Pulse, Negative Going
OE	=	Output Enable (Active Low)
CE	=	Chip Enable (Active Low)
VPP	=	Programming Voltage
VCC	=	Supply Voltage
GND	=	Ground (0V)

22, 26 and 27, and are marked with a "*" next to the pin number.

Pin 26 is not used on the 2764 and A13 on the other devices. Since A13 is outside the addressing range of the 2764, there is no problem leaving it connected to Pin 26 when handling these devices.

SW2 selects which signals arrive at the other four pins. This can be followed through with the circuit diagram and the table above - and does not need a tedious description from me. Note that 2764 and 27128 share the same switch position.

SW3 selects either Program or Read mode. With this switch in the centre (Off) position the EPROM receives 5V power only and can be safely removed from the socket (SK2).

Counters IC5 and IC6 are connected to the EPROM address lines, and are arranged to increment each time a byte of data is received. I will describe the full operation sequence for both Read and Program modes shortly.

IC7 is a tri-state buffer and is enabled in Program mode only. In this mode it connects the received data from the UART to the EPROM. Although the 74LS245 is bi-directional, in this application it operates in one direction only because pin 1 is permanently connected to 0V.

OK, it's time to own up! The circuitry around IC10,Q1 and Q2 is an afterthought. If you look at the internal photo of the prototype you will see the Veroboard (don't worry, the current PCB is correct). I missed the fact that 27256 and 27512 EPROMs need a supply of 6V instead of 5V when programming.

Since there are no spare sections on SW3, I am using the section that connects the programming pulse to pin 20. If the unit is switched to Read, or to Prog and 2764/27128, pin 20 of the socket will be low. If it is switched to Prog and 27256 or 27512 it will be high, except when actually programming in which case it will be high with the 1ms negative going programming pulses. This high level turns on Q1 via R17. Q1 discharges C19, which will not charge significantly during the 1ms periods when Q1 is off. This low level keeps Q2 off, allowing R13 and R14 to bias the GND pin of IC10 to about 1.2V, giving an output of about 6.2V. When Pin 20 is low, Q2 is off and Q1 is on. The GND pin of IC10 is therefore connected to 0V, giving an output of 5V.

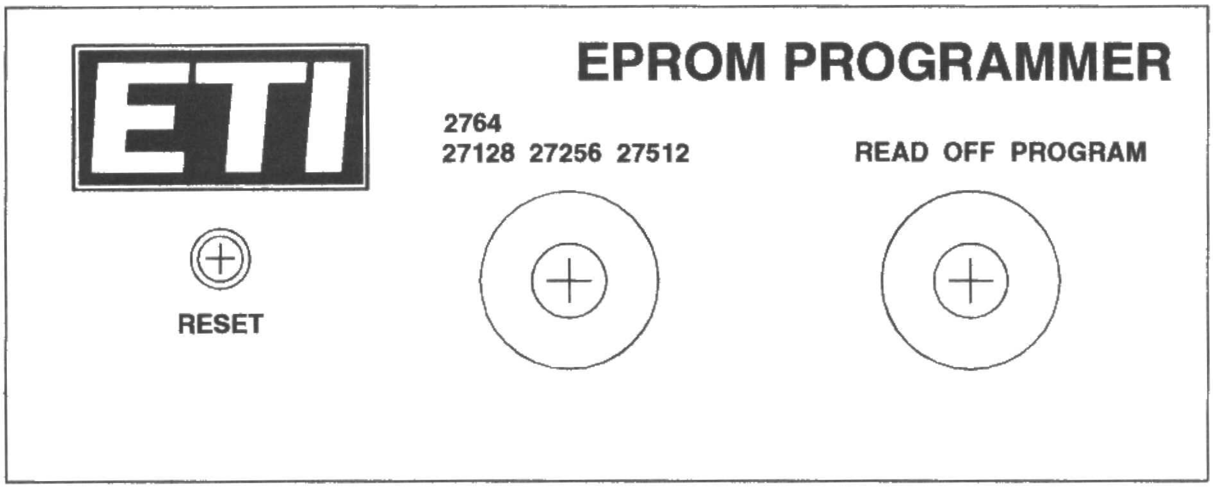
A voltage of 6.2V because some devices need 6V while others need 6.25V; in either case the tolerance is +/- 0.25V. The voltage tends to drop slightly on load due to the varying current from the GND pin of IC10, so I tended towards the high end of the acceptable range. Some EPROMs draw 100mA or more, so a 1A regulator IC was chosen in preference to a 100mA part. IC10 draws its power from the 12.6V programming supply so this should be suitably rated.

Assume that SW3 is in the Read position. When a byte of data is received, it appears on lines R0-R7, and the Strobe- line pulses low as described previously.

Note that in Read mode the actual value of the byte received is irrelevant, since IC7 is disabled. However the receipt of a byte causes the unit to send a byte, which is what we need.

Since the Prog- line is high, pin 13 of IC4 will be held high by D3, so the Strobe- pulse will have no effect in this section of the circuit. This arrangement of diodes on the input of an invertor gate produces a NOR function.

The Read- signal, however, is low, so the Strobe- pulse will



Front panel layout

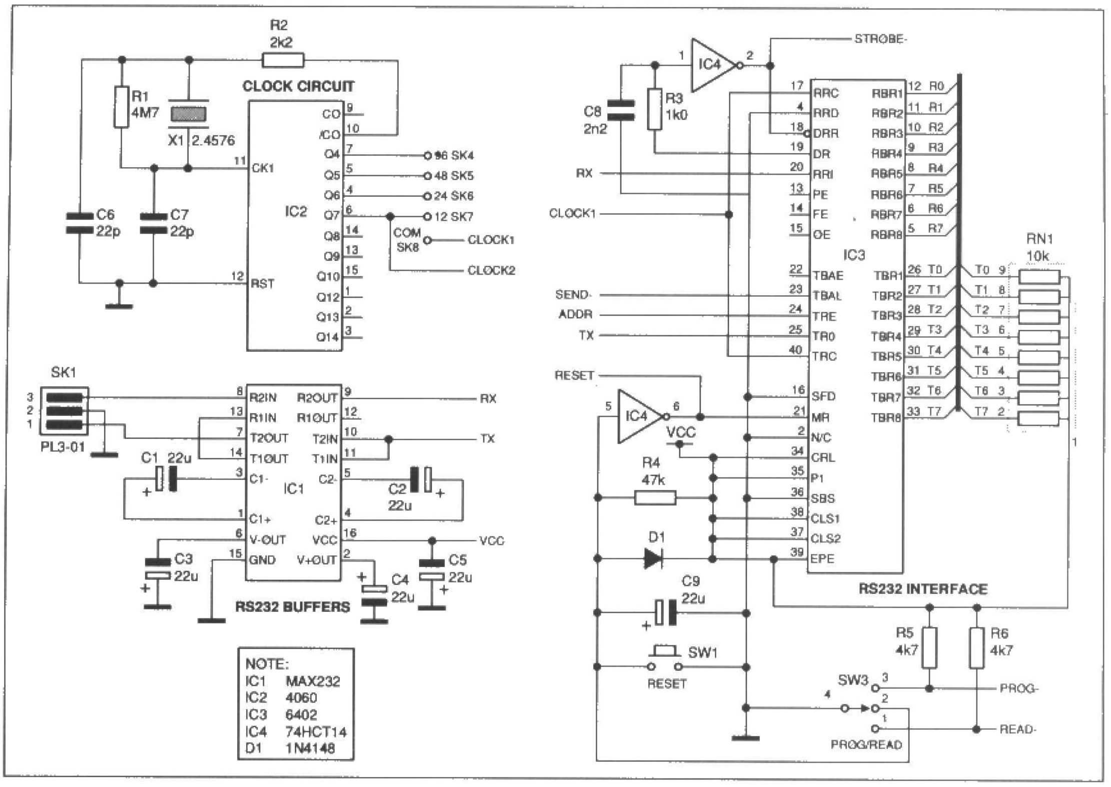


Fig.1. Circuit diagram part 1

pass through D5 and the two gates of IC4, arriving on pin 8. PP- is currently high, so the Strobe- signal will appear inverted on pin 4 of IC4. C14, R11 and D9 act as a basic monostable, giving a very brief negative going pulse on the Send- line as IC4 pin 4 goes low.

This Send- pulse causes the UART (IC3) to transmit the data on its pins 26-33. These pins are connected to the EPROM data pins, so the data transmitted is that contained in the current EPROM address. RN1 holds the lines high if no EPROM is fitted, since IC3 is a CMOS device.

Pin 24 of IC3 will go low once the data is safely in its buffer and will go high again once it has been transmitted. This signal is connected to the clock input of IC5 to increment the address.

The next time a byte is received, the data in the next EPROM memory location will be transmitted, and so on. Providing the user presses the Reset button when requested, the software and programmer will remain in step.

We will now look at the Program mode, where the Prog- line is low. IC7 is enabled due to its pin 19 being low. The data received by the programmer will therefore be coupled to the EPROM data lines.

The situation with our diode NOR gates is now reversed, so the Strobe- pulse will appear on pin 12 of IC4, and not pin 8. Another CR monostable circuit (Q8, D4 and C16) produces a brief negative going Prg- pulse.

IC8 and IC9 generate the 1 millisecond programming pulse. The lower two gates form a S-R flip-flop. When the Prg- pulse occurs, pin 6 of IC9 will go high, causing pin 12 to go low. This takes the reset pin of IC8 low, so that it can count the pulses on its clock input. This pulse train (Clock2) is from the crystal-controlled clock circuit (IC2) and has a frequency of 19.2KHz (2.4576MHz divided by 128), which equates to a period of 52.1uS. When 19 of these pulses have been counted (19 * 52.1uS = 0.99ms), pin 8 of IC9 will go low, reversing the state of the S-R flip-flop and resetting IC8 again.

Thus a 0.99ms negative going pulse will appear on the PP- line. The specification calls for 1ms +/- 5%, so 0.99ms is fine. This PP- signal will go to the appropriate pin on the EPROM, set by SW2. The rising edge of PP- will produce a negative pulse on Send- (via D7), as before. A byte of data will therefore be sent once the programming operation is complete. This will also cause the address counters to be incremented as described previously. The value of the data sent is not important since its purpose is to tell the software that the programming step is done - however, it will be the same as the data received.

Therefore, to program the EPROM, it is simply necessary for the software to send one byte of data, and wait to receive something back before sending the next one.

The use of discrete components to produce other types of logic function from a logic invertor gate may seem odd, but it means I can do the job with one IC instead of four!

Construction

The unit is assembled on a single sided PCB, which is available from the ETI PCB service. The copper track layout is shown in the Foils section at the rear of this magazine, and the component overlay is shown in Figure 4.

A number of wire links are required, which should be fitted before any components, since some pass underneath ICs. I would suggest that the resistors are fitted next, followed by the ICs, then the capacitors, then the remaining parts. Fit a link wire between COM and 96 to set the Baud rate to 9600. Fit SIL header strip or Veropins for the off-board connections.

Fit a 28-way IC socket in the EPROM socket position. Plug three more sockets into this, then plug the ZIF (Zero Insertion Force) socket into the top. If this stack feels unsteady, use some Araldite or similar to hold it together.

The interwiring is shown in the photographs accompanying this article. This should be carried out at this stage, since it is necessary for testing. After testing, the board can be fitted into the case.

The connections for both 9 and 25-way serial connectors, use whatever matches the socket on your computer. On the prototype a 9-way D connector (serial) and a 6-way DIN socket (DC input) were fitted to the case.

The rotary switch connections are shown by giving the pin number or letter marked on the switch body. Please take great care with these, since there are 16 wires to each switch and an error could be difficult to track down. I used coloured ribbon cable, and still managed to get one wrong!

Testing

Connect the unit to a 5V and 12.6V supply. If a test meter is to hand, check for about +9V on pin 2 and -9V on pin 6 of IC1 (MAX232).

Addr Line	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Socket Pin	1	27	26	2	23	21	24	25	3	4	5	6	7	8	9	10

Count	Expected Logic Level															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21845	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
43690	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
65535	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Connect the unit to the serial port on your PC, and run the program "SER-TEST.EXE" on the software disk. When prompted, type "1" or "2" followed by <Enter> to say which serial port you are using. The program does nothing more exciting than wait for you to enter a 2-digit hex number (followed by <Enter> and then sends it to the programmer. It then attempts to read back a number;if it's successful it prints the number, otherwise it prints "**". To exit just press Enter on its own. All the responses in this section will be shown with quotes ("") around them - just type the number (followed by Enter), don't type the quotes.

Connect the programmer to your PC's serial port and a suitable power supply. Switch the programmer to 27512 and Read, and press Reset. Type "00" on the computer, and the software should respond with "FF". Whatever two-digit hex number you type now should bring the response "FF", since the unit is reading the EPROM data lines, which with no EPROM will be pulled high by RN1.

We will now pull one of the data lines low, to give a different reading. Fit a piece of wire between pins 9 and 12 of the EPROM ZIF socket. If you type a number now the unit will return "FE". Leave one end of the wire in pin 12, and connect the other end in turn to pins 10, 11, 13, 14, 15, 16 and 17. Type a number in each case and you should get the following responses - "FD", "FB", "F7", "EF", "DF", "BF" and "7F" respectively. Remove the piece of wire, and switch the unit to Off. Now if you type a number the unit will not respond so the software will show "**".

Now switch the programmer to Program. Type "00", and "00" should be returned. Using a logic probe, test meter or oscilloscope, check the logic levels on pins 9, 10, 11, 13, 14, 15, 16 and 17. They should all be low.

Now type "01". The unit should return "01", and pin 9 should now be high, and the others should remain low. Now enter "02", "04", "08", "10", "20", "40" and "80" in turn. In each case the unit should return the number you entered. After each entry, check the logic levels on the data pins; only one should be high in each case - 10, 11, 13, 14, 15, 16 and respectively.

Switch the unit back to Read, and press Reset. Press Enter on its own to quit the software then run "ADR-TEST.EXE", which is also on the disk. Since the address counters are incremented when a byte is sent, it would take a long time to get the count to 65535 manually! ADR-TEST does it automatically, and pauses at four points to allow you to check the logic levels. Follow the instructions on screen. The table below shows the expected logic levels on the address pins at the four pause points.

If these readings are OK, and the preceding tests were also successful you can be fairly confident that the unit is working correctly. The only sections that weren't checked were some of the device and mode switching. However the likelihood of problems here is remote if you were careful with the switch wiring.

If you have a blank EPROM and a suitable hex file available, you can try programming a device using the information given shortly.

The Case

The prototype was constructed in a plastic case, 190mm * 165mm * 68mm (see parts list for details). A suitable overlay for the front panel is shown in figure *. Two photocopies may be taken (enlarge to 162mm * 64mm), one can then be used as a drilling template while the other may fixed to the front panel with clear, self-adhesive vinyl.

Cut a rectangular hole in the centre of the top for the ZIF socket, then mount the PCB on suitable spacers. The sockets for DC power input and RS232 input are fitted on the rear panel.

Software

The software for this project is supplied on one 3.5" 720K disk which is available from the author. Please note that this disk also contains the software for a matching EPROM Emulator, planned for publication next month.

Since the various programs extend to over 2500 lines of BASIC source code, it would make boring reading to print it in the magazine. A printed listing is not available since it would be more expensive than a disk to produce.

A batch file is supplied on the disk to simplify installation.

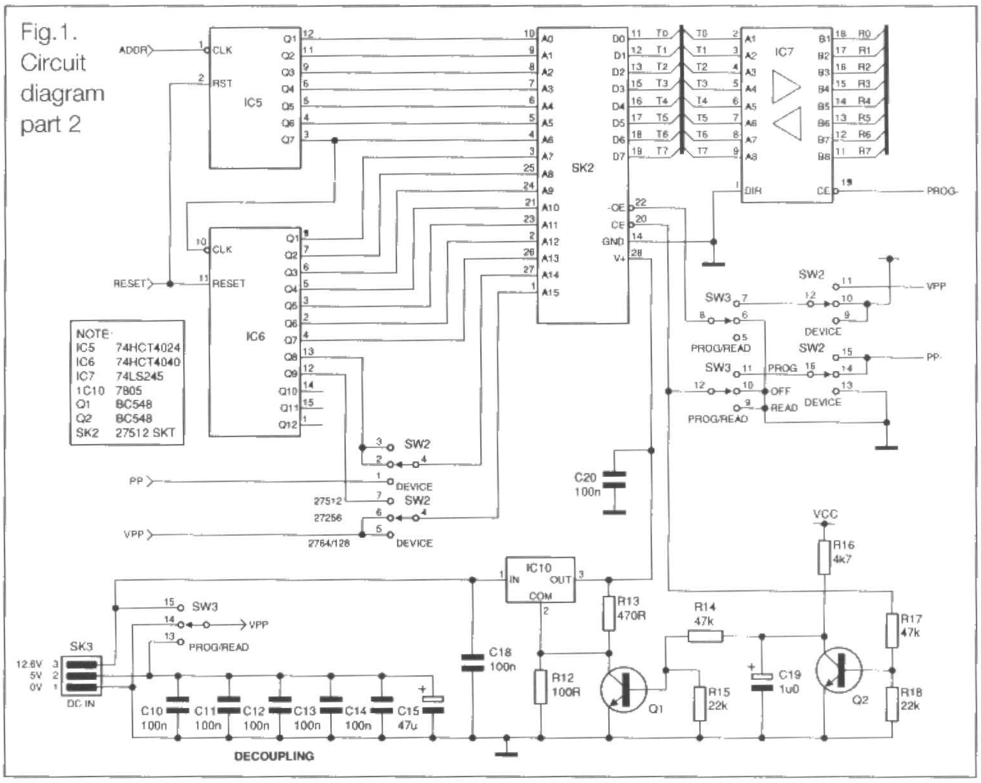


Fig.1. Circuit diagram part 2

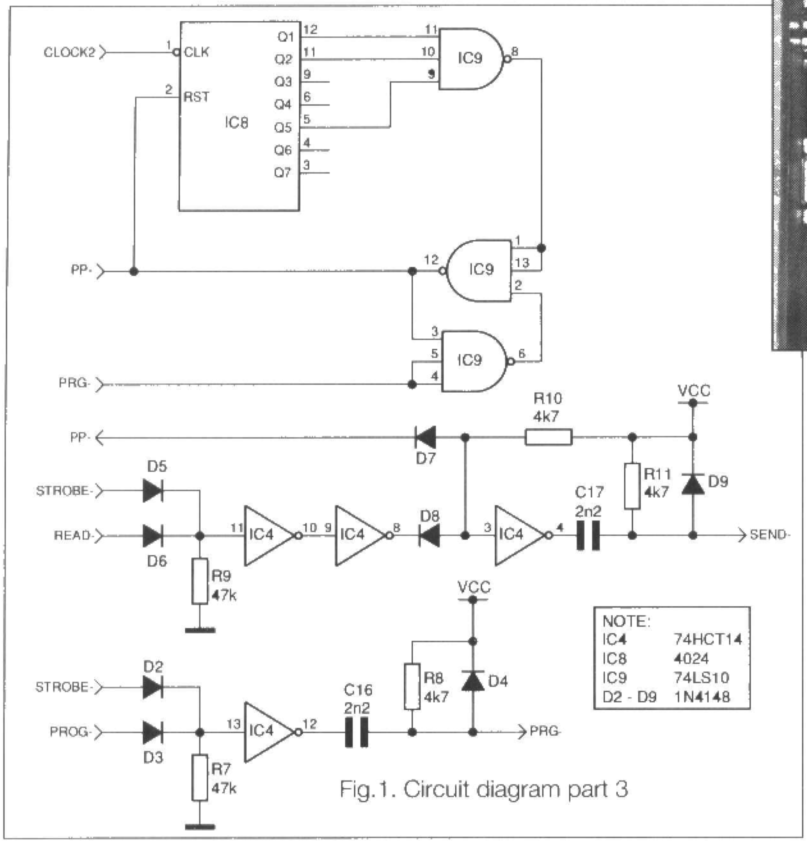
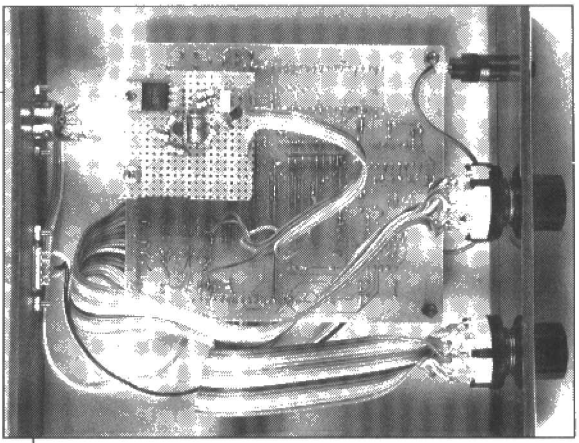


Fig.1. Circuit diagram part 3



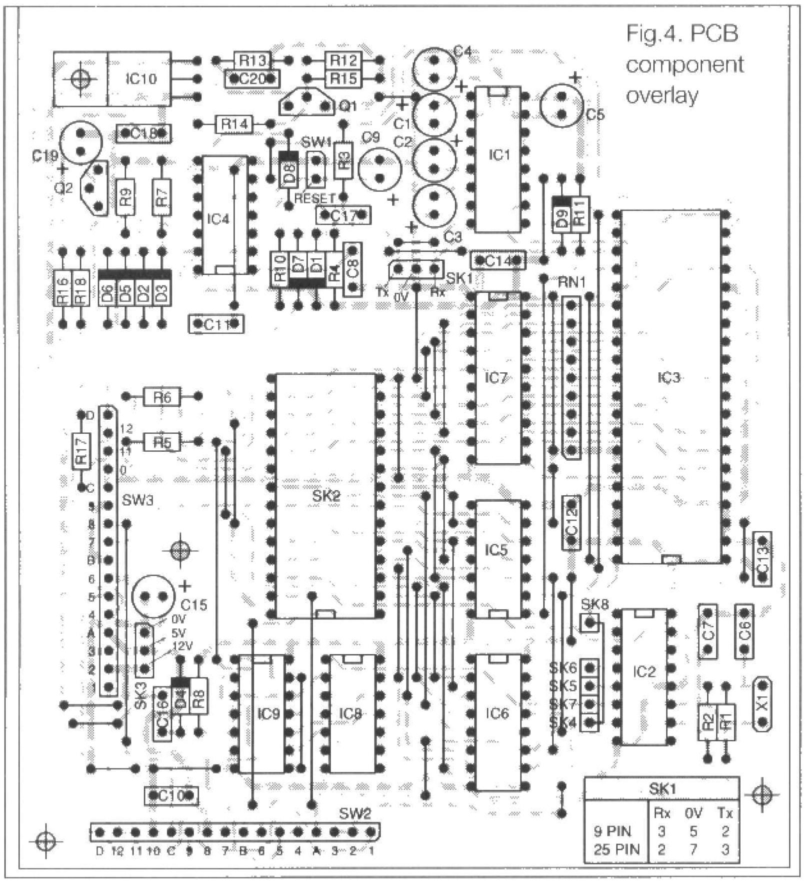


Fig.4. PCB component overlay

Insert the disk in the drive, type "A:" then "INSTALL", and the batch file will make a \EPROM directory on your drive C:, and copy the software to there. If you do not have a hard disk, make a working copy of the disk using DISKCOPY, then put the original away. Do not write-protect your working copy or the software will not work.

If you are using Windows, suitable icon, PIF and group files are supplied on the disk. Some parts of the software will operate much slower under Windows, particularly the initialisation when the serial port is opened. However it will run in the background (probably very slowly) if you are using 386 Enhanced Mode.

The main software of interest for this project is spread over two programs, "PROGRAM.EXE" and "HEX-CONV.EXE". The first of these is the main control software for the programmer, while the second converts various industry standard hex file formats to and from the EPROM programmer format. Formats supported include Intel, Motorola S-Record (3 variants), Tektronix, RCA Cosmac, Binary Image, ASCII Hex (3 variants) and ASCII Binary (4 variants) - so you should be able to find something suitable!

Additional programs on the disk are "EMULATE.EXE" which controls next month's EPROM Emulator, and "SPLIT2.EXE" & "SPLIT4.EXE" which divide Intel hex files into 2 or 4 files for 16 and 32 bit systems respectively. Since the full BASIC source code is given for all of these programs, it would be possible to create one large program containing all the facilities - if someone had more time than me! I would be interested to see any enhanced versions.

When "PROGRAM.EXE" is started, the Device Selection Menu will appear. From here you choose the type of device you will be using, either 2764, 27128, 27256 or 27512. CMOS versions of these devices can also be handled, so if you are using a 27C128 you would choose 27128.

One important point regarding programming voltages. Early 2764 and 27128 devices required a programming voltage of 21V. If the device is marked 27C64, 27C128, 2764A or 27128A it will require 12.5V. Any non-CMOS device without an 'A' suffix will require 21V. If in doubt, try programming with 12.5V first; no damage will occur but the device will not be programmed. If a 12.5V device is programmed with 21V it will be destroyed. 21V devices can be programmed by this unit, but you will need to provide an appropriate voltage, and a heatsink for IC10.

Once you have chosen the device required, the Main Menu will appear. Option 1 allows you to read an EPROM, and save the data to disk. The data is saved and loaded ASCII-Text format which is peculiar to this software. "HEX-CONV.EXE" will convert to and from this format.

When Option 1 is selected you will be told where to set the switches, and to insert the EPROM to be read. You will then be asked for a filename; simply enter eight alpha-numeric characters - the extension is fixed to .HEX and does not need to be typed. If a file of that name already exists, it will be overwritten. Now sit back and wait; the progress will be shown on the screen.

Option 2 allows you to check an EPROM is blank. A blank device will have "FF" in all locations. The operation is the same as above but no filename is needed. Note that if no device is fitted, the blank test will pass since the data lines are pulled high by RN1.

An EPROM can only be erased by exposing the window in the case to ultra-violet light. If a proper eraser is used this will take about 20 minutes. Replacement tubes for these are available for about £10 and will fit into some fluorescent torches - leave out any clear plastic or glass in front of the tube since this will block UV. Be very careful not to look at this light.

I have not yet found a completely successful alternative. A disco type 2' UV fluorescent tube will erase a device in about two days! I have been told that a camera flash gun will erase a device in about 4 flashes, but it did not work with my point-and-shoot camera. A professional flash unit will probably be rather more successful. If you know of a successful DIY erasing method, please write to the letters page and share it with the rest of us!

Option 3 allows you to program an EPROM from data on disk. Operation is the same as above, but the programming process will take about twice as long as reading. Once the programming is complete, you will be asked if you wish to program the device.

Option 4 allows you to verify an EPROM. This checks the EPROM against a file on disk, and is normally done after programming to make sure the programming was successful.

Option 5 allows you to change the EPROM type as previously. Option 6 lets you run the Hex File Convertor program, "HEX-CONV.EXE" - this is detailed below. Option 7 lets you access a DOS Shell, type "EXIT" to return to the programmer. To quit the Programmer, press Escape.

The Hex File Convertor is equally simple to use. When it is started, you get the Load Menu. Choose the file format of the existing file, then enter the filename when requested. As previously, the file extension is .HEX and need not be typed.

Once the file is loaded, the Save Menu will appear. Choose the file format you want, then type the filename. If a file of that name already exists it will be overwritten without warning - so be careful!

The file "HEX-CONV.TXT" gives information on all the file formats, and samples are provided on the software floppy disk, in the \HEX directory.

(IMPORTANT: The software for the EPROM programmer and emulator has been thoroughly tested, but please note that it is supplied as-is, and neither Paul Stenning or Electronics Today International can accept any liability for any loss or damage, however caused. The source code is supplied so that you may modify the software for your own use only. The software may not be redistributed in either its original or in a modified form. If you cannot accept these conditions please do not order the software disk.)

Conclusion

Elsewhere in this issue you will find a suitable power supply for this unit. It will supply 5V at up to 500mA and 12.6V at up to 250mA. This will also power next month's EPROM Emulator.

Coming next month is a matching EPROM Emulator. This emulates the same range of devices as the programmer, and can save hours programming and erasing EPROMs when doing software development work. It needs a single 5V supply and can be powered from the host circuit (if it can spare 100mA) or from the power supply mentioned above.

Parts

The plastic case used for the prototype is made by Bafbox, and is available from RS/Electromail, stock no 506-788.

The software for this EPROM Programmer and next month's EPROM Emulator are available on one disk from the author. Please send a blank PC formatted 720K 3.5" disk, the price for the software is £10, If you do not have a suitable disk, send £12 and one will be supplied. Please add an extra £1 if within Europe, or £2 elsewhere, to cover the additional postage.

The author can also supply PCBs for the EPROM programmer, the PSU, and next month's project the EPROM emulator. The EPROM PCB is priced at £12, the PSU PCB at £5, and the Emulator PCB at £15, or if you want to buy all three then they are available at £30. Do not forget to add P&P: £1.50 in the UK and £2.50 elsewhere.

If you would like to purchase either the software or the PCBs then send a cheque or postal order for the appropriate amount made payable to "Paul Stenning", plus a return address label to the following address:

Paul Stenning,1 Chisel Close,Hereford,HR4 9XF

If ordering from outside the UK, please make sure your cheque is in Pounds Sterling and is drawn on a UK bank. The author will be making up PCBs in batches so please be patient; cheques will not be cashed until the boards are dispatched.

The above offers from the author are only valid until the end of March 1995,;after that date please contact him first before sending any cash.

PARTS LIST

Resistors (0.25W 5% or better)

Q1	4M7
Q2	2K2
Q3	1K0
Q4,7,9,14,17	47K
Q5,6,8,10,11,16	4K7
Q12	100R
Q13	470R
Q15,18	22K
QN1	10K x 8 Resnet

Semiconductors

IC1	MAX232
IC2	74HCT4060
IC3	6402
IC4	74HCT14
IC58	74HCT4024
IC6	74HCT4040
IC7	74LS245
IC9	74LS10
IC10	7805
TR1,2	BC548
D1,2,3,4,5,6,7,8,9	1N4148a

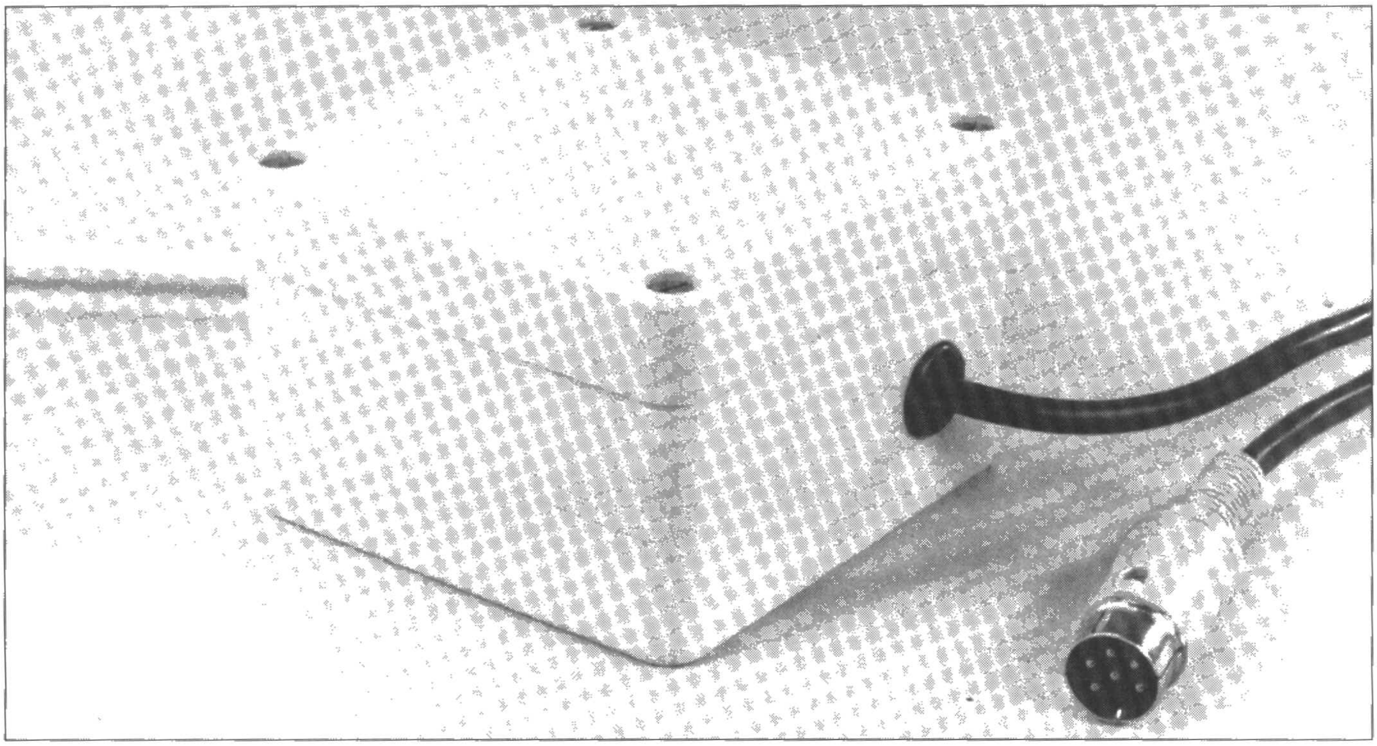
Capacitors (all 16V or greater)

C1,2,3,4,5	22u
C6,7	22p
C8,16,17	2n2
C9	10u
C10,11,12,13,14,18,20	100n
C15	47u
C19	1u0

Miscellaneous

XT	2.4576MHz
SK1	9 or 25 way D
SK2	28 way ZIF socket
SK3	DC INPUT
SW1	push to make
SW2,3	4 pole 3 way rotary
PCB, Case, knobs, wire.	

EPROM PROGRAMMER AND EPROM EMULATOR POWER SUPPLY



This handy little power supply unit was designed by Paul Stenning for his EPROM programmer and emulator projects, but could be used for a wide range of different applications.

This simple power supply unit was designed for use with the EPROM Programmer, featured elsewhere in this issue. It is equally suitable for powering the matching EPROM Emulator which will appear next month.

We are printing this as a separate article to avoid confusion with component reference numbers. The unit produces 5V at up to 500mA, and 12.6V at up to 250mA. At currents over about 250mA from the 5V rail, a heatsink will be needed for IC1 - this is not needed if the unit is used with the projects listed above.

How it Works.

The circuit is shown in Figure 1. Transformer X1 produces 6V AC from the mains input. This is rectified by D1 and D2, and smoothed by C1 giving about 8.5V DC. IC1 is a standard three-pin 5V regulator, which does just that!

For the 12.6V supply, we have used a voltage doubler circuit (D3, D4, C5 and C6)

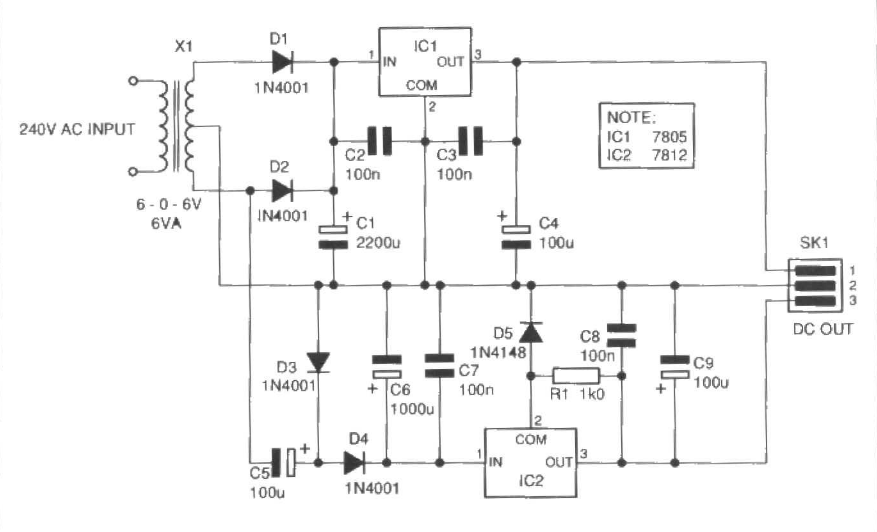
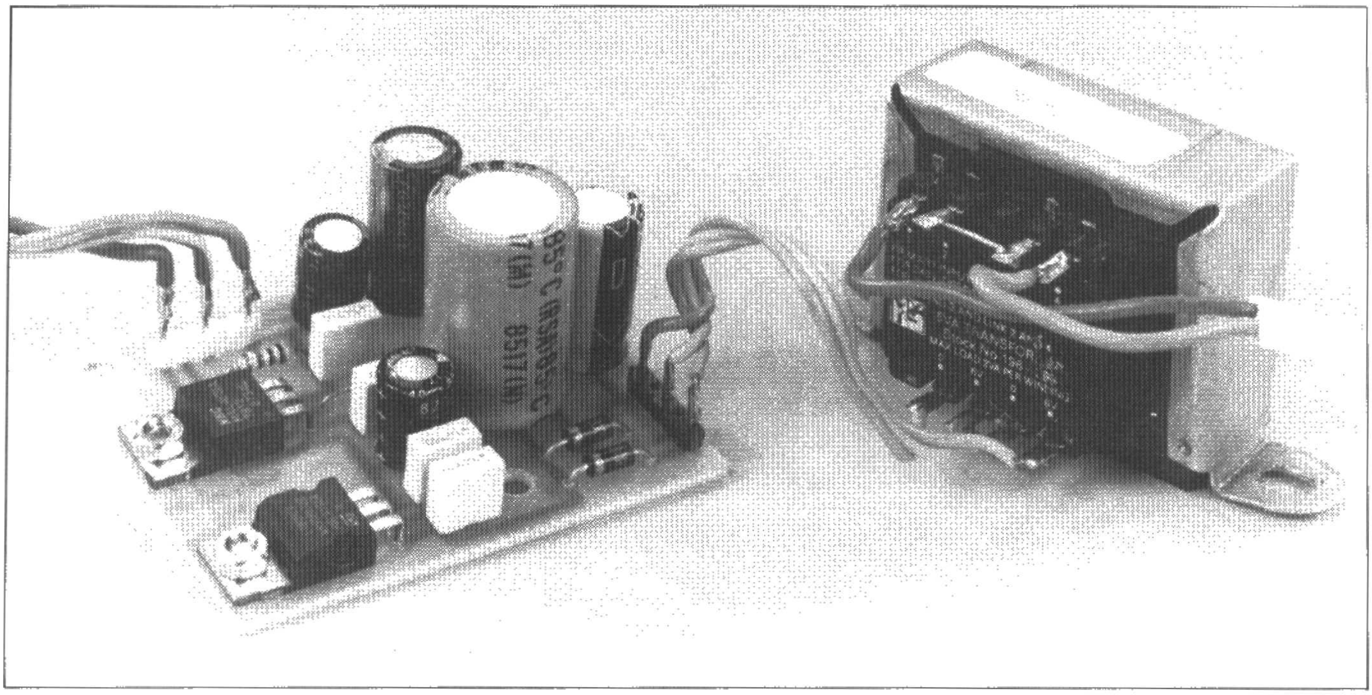


Fig.1 PSU Circuit Diagram



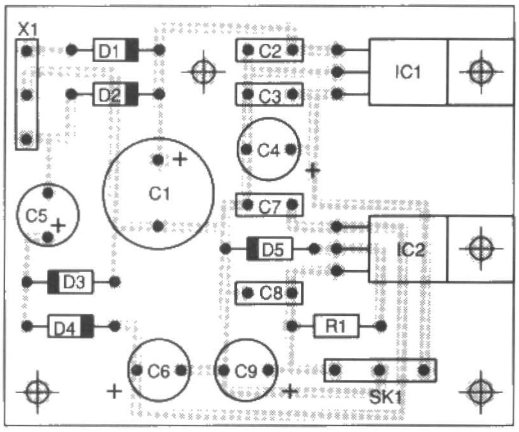


Fig. 2 PSU conponent overlay

PARTS LIST

IC1	7805
IC2	7812
D1,2,3,4	1N4001
D5	1N4148
R1	1K0
C1	2200uF 16V
C2,C3,C7,C8	100nF
C4,C9	100uF 16V
C5,C6	1000uF 25V
SK1	DC OUTPUT
X1	6-0-6V 6VA

Case, PCB, cable, 13A plug with 3A fuse.

giving about 16V. This is regulated by a 12V regulator, with a diode (D5) in the ground lead to increase the voltage by 0.6V. EPROMs require either 12.5V or 12.75V with a tolerance of 0.25V for programming. 12.6V will comply with either of these requirements.

A 6V transformer was used to reduce the power dissipation in the 5V regulator, IC1. A voltage doubler circuit operates very well when a lower current is required, as is the case with the 12.6V rail.

Construction.

All the components, except the transformer, are mounted on a small PCB, the foil for which is shown in the Foils section at the back of the magazine (the author can supply ready-made PCBs; see the end of the EPROM programmer project article for details). The component overlay is shown in Figure 2.

No heatsinks are required for the intended use. If you are likely to use the unit for other purposes, it would be a good idea to fit a heatsink on IC1.

The prototype was fitted in a small plastic box, 80 x 80 x 55mm. This was rather a tight squeeze, so something a little larger is recommended.

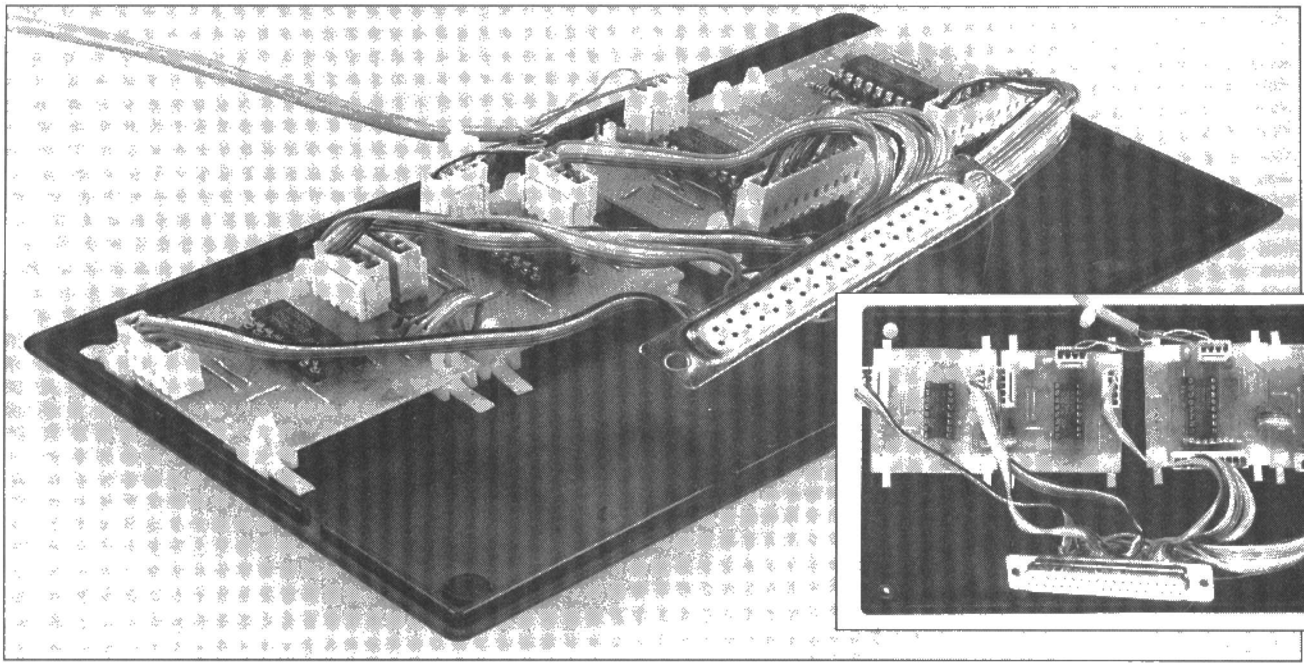
The mains input arrives via a length of 2 core mains flex. The DC output is connected to a 6-pin DIN plug via a length of 3-core mains flex. A matching 6-pin DIN socket is fitted on the EPROM Programmer and other projects.

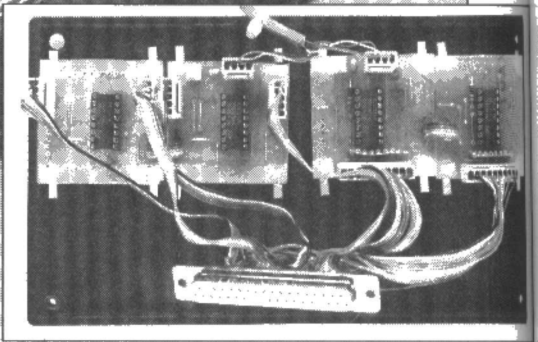
NEXT MONTHS

This power supply will be used in next months Eprom, Emulator project

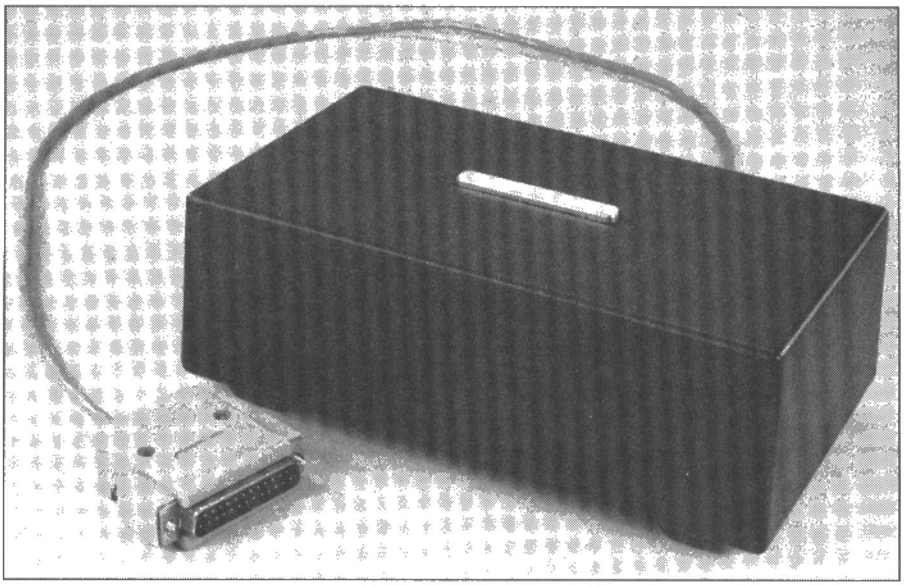
PROJECT

Parallel MULTI-PORT



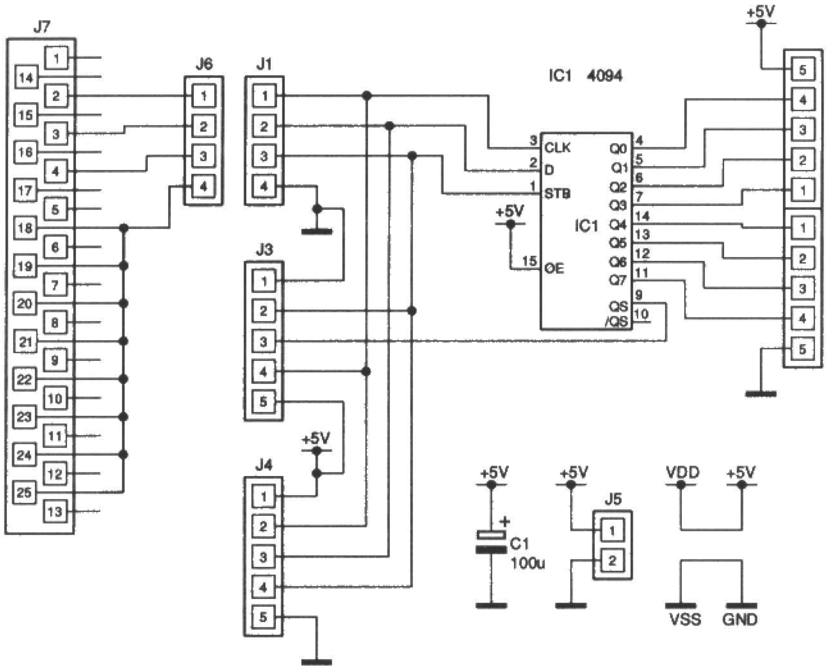


In this project Tristan Grant shows how to expand the parallel port on a PC to provide an almost unlimited number of 8 bit Input or Output Ports



The standard method of equipping a personal computer with multiple Input or Output ports is to use a plug-in card that fits into the expansion slots in the PC's motherboard. However, many users of either notebook computers or standard PCs that have no space for extra, big Input/Output (I/O) cards will realise how difficult it can be to carry out I/O functions. I had been experiencing this problem on my notebook, and as a result I designed this unit, which can carry out the low speed control tasks that I require. It also allows for easy sharing of the I / O ports that it provides because it is very simple and fast to disconnect from one computer and connect to another.

The unit presented here will allow a PC equipped with a parallel port to have an almost unlimited number of 8 bit Input or Output ports. The number of ports attainable is limited only by the power supply available, the drive capabilities of the parallel



J1, J6 - 4 WAY HEADER

1	D0 FROM PC (CLOCK)
2	D1 FROM PC (DATA)
3	D2 FROM PC (LOAD)
4	GND FROM PC

J2A - 5 WAY HEADER

1	+5V
2	Q0
3	Q1
4	Q2
5	Q3

J2B - 5 WAY HEADER

1	Q4
2	Q5
3	Q6
4	Q7
5	GND

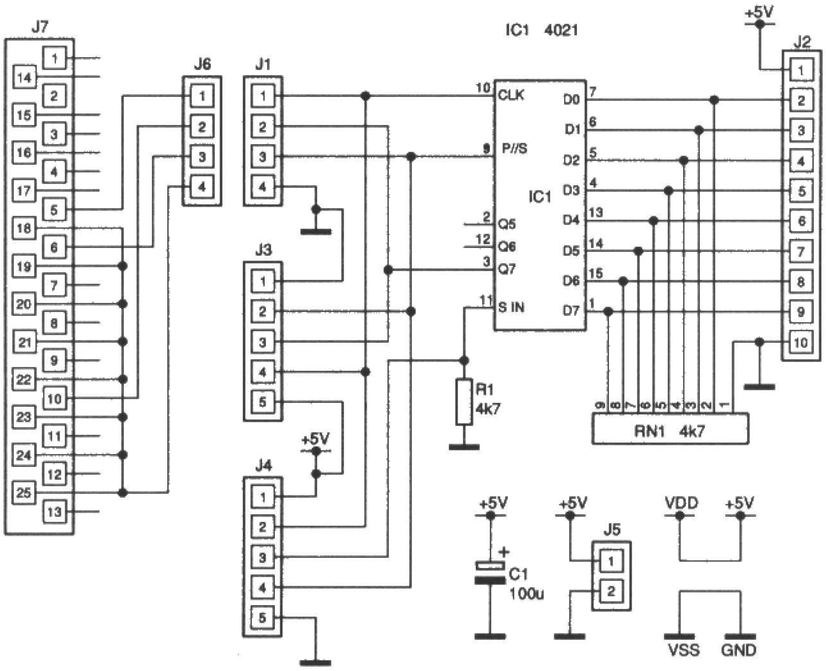
J3 - 5 WAY HEADER

1	GND
2	STROBE
3	QS
4	CLOCK
5	+5V

J4 - 5 WAY HEADER

1	+5V
2	CLOCK
3	DATA
4	STROBE
5	GND

Fig.1. Output circuit



J1, J6 - 4 WAY HEADER

1	D3 FROM PC (CLK)
2	ACK FROM PC (OUT Q7)
3	D4 FROM PC (P/S)
4	GND FROM PC

J2 - 5 WAY HEADER

1	+5V
2	D0
3	D1
4	D2
5	D3
6	D4
7	D5
8	D7
9	D7
10	GND

J3 - 5 WAY HEADER

1	GND
2	P/S
3	Q7
4	CLOCK
5	+5V

J4 - 5 WAY HEADER

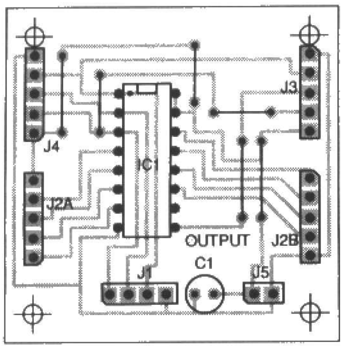
1	+5V
2	CLOCK
3	S IN
4	P/S
5	GND

Fig.2. Input circuit

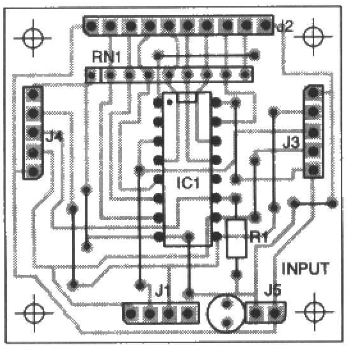
port and by the minimum speed at which you are prepared to work. The more ports that you have, the slower the system will be; on the other hand it is preferable to have some slow I/O ports that can do useful work instead of having no I/O capability at all.

As you will have gathered by now, the parallel port is used to interface the unit to the computer. The only signals required are some of the data lines used to provide control and data signals and the ACK (Acknowledge) line which is used for data input from the unit.

My primary aim for the unit was that its operation should be as simple as possible, that there should be a low chip count in order to keep the cost down and expansion of input or output channels should be as easy as possible.



Output component overlay



input component overlay

Operation of the Circuits.

As you can see from Figs 1 and 2, the Input and Output circuits are extremely simple; only a single shift register is required for each one, along with a few discrete components and several connectors. I will deal with the method of operation of each of these circuits in turn.

First though, it is worthwhile to give a brief description of the operation of a shift register.A shift register is basically a series of flip flops connected so that the Q output of one flip flop drives the D input of the next flip flop. If all of the clocks are driven simultaneously then data, entered at the D input of the left-most flip flop, will shift one place to the right with each clock pulse and is made available at the Q outputs of the flip flops. This makes them particularly suitable for serial to parallel conversion as used in the Output section of the unit.

If the flip flops are connected in a different way, such that the parallel data presented is used to set the flip flops' Q outputs to the value of the data, then it is possible to convert parallel data into serial data. This technique is used for the Input section of the unit.

The output circuit operates in the following manner: IC1 is a 4094 serial to parallel converter. This is connected to the parallel port of the PC so that D0 controls the CLK line, D1 provides the data to be clocked into the flip flop and D2 loads the data from the flip flops into the output latches of the shift register. Data presented at the DATA input of IC1 is clocked into the flips flops on each positive-going pulse presented at the CLK input of IC1. When data has been clocked into the shift register, the STB input is pulsed high, briefly. This loads the data into the output latches of IC1. It is possible to disable the outputs of IC1, but in this application that feature was thought to be unnecessary, so the outputs are permanently enabled by connecting OE to +5v.

Expansion of output modules is carried out by applying the QS output of the first shift register to the DATA input of the second shift register. All of the other connections, STB and

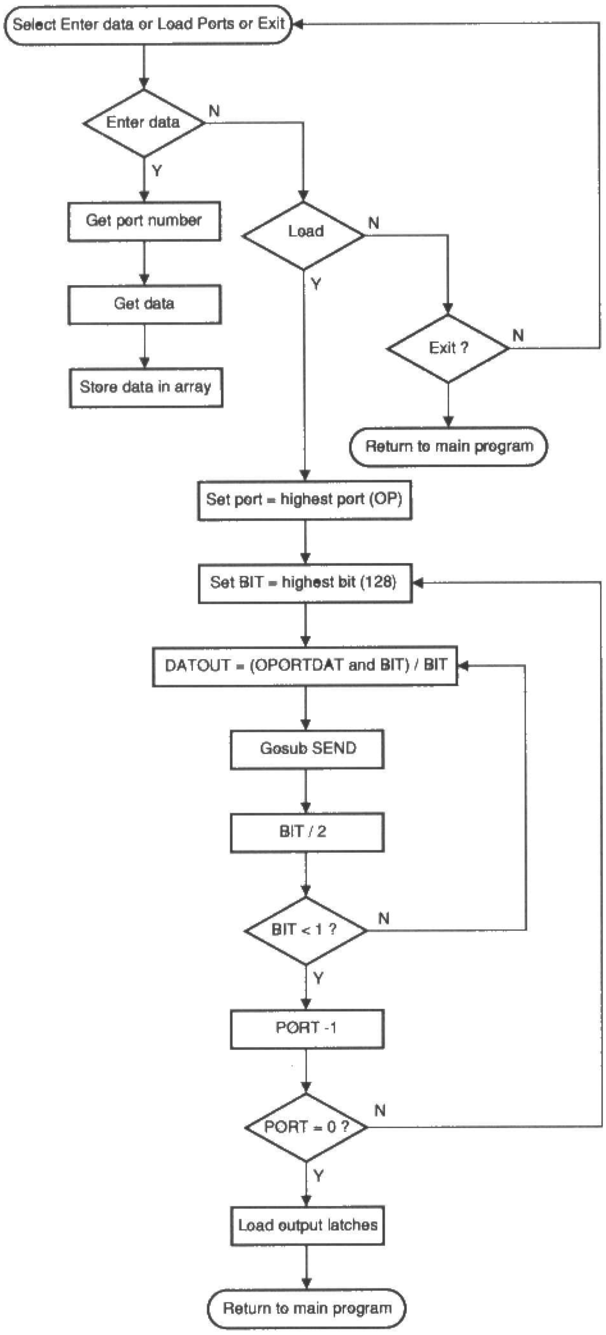


Fig.3. Flow diagram of data outputting routine (OUTPUT DATA)

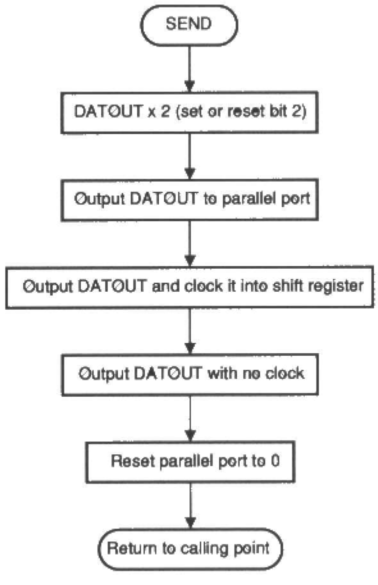

SEND
DATOUT x 2 (set or reset bit 2)
Output DATOUT to parallel port
Output DATOUT and clock it into shift register
Output DATOUT with no clock
Reset parallel port to 0
Return to calling point

Fig.4. Flow diagram of data sending routine (SEND)

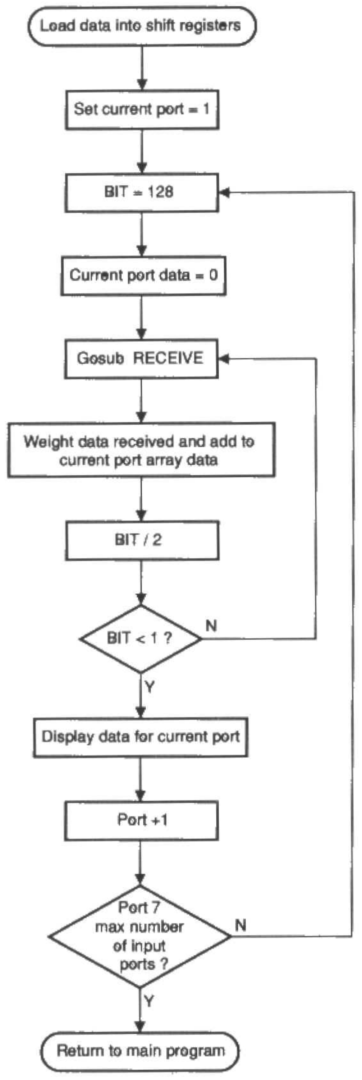

Load data into shift registers
Set current port = 1
BIT = 128
Current port data = 0
Gosub RECEIVE
Weight data received and add to current port array data
BIT / 2
BIT < 1 ?
N
Y
Display data for current port
Port +1
Port 7 max number of input ports ?
N
Y
Return to main program

Fig.5. Flow diagram of routine to input data from ports (INPUT DATA)

CLK, are daisy-chained onto the respective connections of the next shift register. The first port always has the lowest identity. Data transmission from the PC is ordered so that the most significant bit of the highest port comes first and the least significant bit of the first port comes last.

The operation of the input circuit is as follows: IC1 is a 4021 parallel to serial converter. It is connected to the parallel port so that CLK is connected to D3, P/S (load) is connected to D4 and Q8 (serial data out) is connected to ACK. Data presented at the data inputs of IC1 is loaded when P/S is pulsed high; pulsing CLK high will then cause the data to shift through the register and be presented at Q8. Resistor network RP1 is used to stop the inputs floating high if there is no signal connected.

Expansion of Input modules is carried out by connecting the SIN (serial in) connection of the first module to the Q8 OUT connection of the second module. All of the other control connections, P/S and CLK, are daisy-chained onto the respective connections of the next shift register. As with the Output modules, the first port always has the lowest identity. Data transmition to the PC is ordered so that the most significant bit of the lowest port is sent first and the least significant bit of the highest port is sent last.

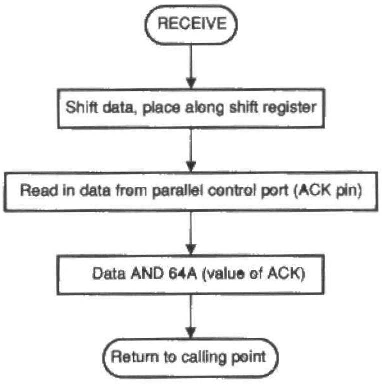

RECEIVE
Shift data, place along shift register
Read in data from parallel control port (ACK pin)
Data AND 64A (value of ACK)
Return to calling point

Fig.6. Flow diagram of data receiving routine (RECEIVE)

Construction of the Unit.

The unit can be made up from as many Input or Output boards as required; the two boards have been produced separately because the same number of output and input boards might not be required. Producing both circuits on one board would increase the cost of some implemenations. Construction of the PCBs shouldn't cause any problems. It is advisable to drill any mounting holes in the boards before soldering in the components. I would very strongly recommend that you insert the wire links first, especially the one on the input board that is partly under IC1, otherwise you will have to mount it on the underside of the board. Also, the link between the resistor network RP1 and connector J2 on the input board should be inserted at the start as working space becomes rather tight.

The ICs should be mounted in sockets to save de-soldering in case they become damaged and have to be replaced. I prefer to use the strip connectors that can be snapped off to the required size, as they give a good clearance over one of the wire links. They provide for a good connection with the pins on the IC. Connection to the PC printer port is by a 25-way D type male connector with solder bucket terminals. Be sure to short pins 18 to 25 together, as the ground on some printer ports isn't always where you would expect.

The interface cable will need to be eight cores if both types of board are to be used together and should be screened. It is preferable not to exceed a length of 2 metres for the cable. The connectors on the unit are the standard 0.1" pitch locking type or, if you want to save money and don't anticipate having to disconnect daisy-chained boards, then use ribbon cable to

PARTS LIST

OUTPUT BOARD

CAPACITORS.	
C1	100nF
Integrated Circuits.	
U1	4094
MISCELLANEOUS.	
J1	4-way header
J2A	5-way header
J2B	5-way header
J3	5-way header
J4	5-way header
J5	2-way header
Input Board	
CAPACITORS.	
C1	100nF
Resistors.	
RP1	4K7 Ohm 8 way SIL
R1	4K7 Ohm
Semiconductors.	
U1	4021
MISCELLANEOUS.	
J1	4-way header
J2	10-way header
J3	5-way header
J4	5-way header
J5	2-way header

INPUT BOARD

CAPACITORS.	
C1	100nF
RESISTORS.	
RP1	4K7 Ohm 8 way SIL
R1	4K7 Ohm
SEMICONDUCTORS.	
U1	4021
MISCELLANEOUS.	
J1	4-way header
J2	10-way header
J3	5-way header
J4	5-way header
J5	2-way header

Listing 1.

```
REM parallel.bas
REM multi-channel I/O  via PC Parallel Port using shift registers
REM demonstration control software
REM T.B.Grant 19 / 6 / 1994
REM check that the base address of your parallel port is the same as the REM one given for 'parallelop'
parallelop = &H3BC:          REM parallel port data output address
                     REM  &H3BC for XT/MDA type ports,
                     REM  &H378 for AT type ports,
                     REM  &H278 for AT second port
parallelip = parallelop + 1:  REM parallel port data input address
clkoh = 1:                  REM output shift register CLK high
clkol = 0
loadoh = 4:                 REM output shift register STB high
loadol = 0
shiftih = 8:                REM input shift register CLK high
shiftil = 0
loadih = 16:                REM input shift register P/S high (load)
datail = 0
mask = 64:                  REM value of ACKnowledg

REM reset parallel port
control = 0
OUT parallelop, control
CLS
REM get the numbet of ports
  LOCATE 5, 1: INPUT "Number of 8 bit output ports :", op
  LOCATE 6, 1: INPUT "number of 8 bit input ports  :", ip
REM set up arrays for port data
 DIM oportdat(op)
 DIM iportdat(ip)
CLS
REM main menu routine
menu:
CLS
LOCATE 1, 15: PRINT "Multi chanell I/O from PC parallel port"
LOCATE 4, 1: PRINT "Enter 1 to set up the system"
LOCATE 5, 1: PRINT "Enter 2 to output data"
LOCATE 6, 1: PRINT "Enter 3 to input data"
LOCATE 7, 1: PRINT "Enter 4 to quit"
LOCATE 8, 1: INPUT "Selection:", choice
IF choice = 1 THEN
 CLS
    ERASE oportdat:               REM erase old arrays
    ERASE iportdat
    LOCATE 5, 1: INPUT "Number of 8 bit output ports :", op
     LOCATE 6, 1: INPUT "number of 8 bit input ports  :", ip
     DIM oportdat(op):            REM declare new arrays
     DIM iportdat(ip
   ELSEIF choice = 2 THEN
    CLS
    GOSUB outputdata:               REM goto routine to output data
   ELSEIF choice = 3 THEN
    CLS
    GOSUB inputdata:               REM goto routine to input data
   ELSEIF choice = 4 THEN
    GOSUB finish
   ELSE LOCATE 20, 1: PRINT "Not a valid entry"
  END IF
  GOTO menu
REM routine to output data
outputdata:
CLS
LOCATE 20, 1: INPUT "1 to enter data, 2 to load ports, 3 to exit menu"; choice
```

```
  IF choice = 1 THEN
redo:
       LOCATE 21, 1: INPUT "Enter port number :"; number
         IF number > op THEN
           GOTO redo
         END IF
      LOCATE 22, 1: INPUT "Enter data        :"; datent
        oportdat(number) = datent
      GOTO outputdata
    ELSEIF choice = 2 THEN
      GOTO loading
    ELSEIF choice = 3 THEN
      GOTO menu
    ELSE
   LOCATE 23, 1: PRINT "Invalid entry, try again!!"
      GOTO outputdata
  END IF
loading:
 FOR port = op TO 1 STEP -1:             REM carry out for each 8bit port
   bit = 128
    DO UNTIL bit <1:             REM test bits to produce serial data
     datout = oportdat(port) AND bit
      datout = datout / bit
      GOSUB send
       bit = bit / 2
    LOOP
 NEXT port
REM latch new data into shift register outputs
 control = loadoh
  OUT parallelop, control
 control = loadol
  OUT parallelop, control
RETURN menu
REM procedure to send data to the output sh1ft registers
send:
 control = 0
  datout = datout * 2:    REM put data into correct position in control word
   control = datout
   OUT parallelop, control
   control = datout + clkoh
    OUT parallelop, control
   control = datout
    OUT parallelop, control
   control = 0
    OUT parallelop, contro
RETURN
REM procedure to read in data from the parallel port
inputdata:
REM load data into the shift registers
 control = loadih
  OUT parallelop, control
   control = loadil
    OUT parallel, control
FOR port = 1 TO ip STEP 1
 bit = 128
  ipportdat(ip) = 0
   DO UNTIL bit < 1
     GOSUB receive
REM accumulate the data bits in the current input port array
      ipportdat(ip) = ipportdat(ip) + (bit * datin)
       bit = bit / 2
   LOOP
REM print value of data present at port
   LOCATE (4 + port), 1: PRINT "Port";
```

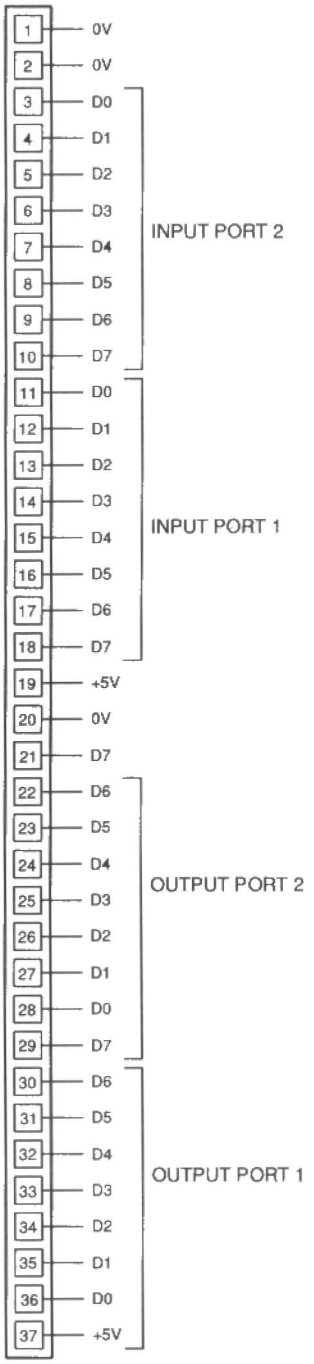


Fig.7. Connections of 32-way connector

```
    PRINT port;
    PRINT " :=";
    PRINT ipportdat(ip);
    NEXT port
    LOCATE (4 + port + 1), 1: INPUT "Any key to continue", dummy
  RETURN menu
  REM procedure to check whether ack set to one and increment shift register
  receive:
   control = shiftih
    OUT parallelop, control
   control = shiftil
    OUT parallelop, control
     ackin = INP(parallelip)
  REM check if ack line set (bit 8 high) set ie data 1 from shift register
   datin = (ackin AND mask) / mask
  RETURN
  finish:
  END
```

hard wire the boards together. Only the first input and output boards need to have an external power supply (+5v) connected to them, as this is passed on to the other boards by the daisy-chaining connection. On most boards, only Connectors J1 to J4 will be required. Don't forget that on the first input board, connector J3 can be omitted and on the first output board, connector J4 can be omitted.

The connector type used for connecting external circuits to the unit is up to the constructor; different applications will require different connectors. However, my preference in this case was to use a 37-way D type connector. When you have finished the boards check very carefully for any solder bridges and check also that the link between the 25W D type connector on the parallel port and the relevant connectors on the input and ouput boards is correct.

Interconnection of boards.

To link together multiple input or output boards, you must follow this sequence. For output boards, link connector J3 on the first board to connector J4 on the second board; I have designed the cicuit so that the cable between the boards does not have to be twisted. For input boards, link connector J4 on the first board to connector J3 on the second board. As I mentioned earlier, you will only need to supply power to the first input and output boards as power will be distributed amongst the boards by the daisy-chaining connectors.

Software Description.

A disadvantage of the simple hardware is that a relatively bulky piece of software is needed to control it. However, it should be reasonably easy understand.The control program shown in listing 1 is very much intended as a demonstration piece to show how the ports are controlled and allow for fast development of your own special application software.

Those of you who have been following the design closely will realise how the parallel port is used to control the external circuits. To provide a clear explanation, look at the flow charts shown in Figs 3 to 5, each of which shows the operation of the main input and output routines, and compare these with the relevant routines in the program. If you want to speed up the operation of the unit, then I would suggest one of these options: either compile the code before running it, or modify the code and add assembler routines to speed up the data output to the parallel port, or write the code in a language that has an efficient compiler; Pascal would probably be ideal for this.

Using the Software.

Before running the software, check that the base address of the parallel port that you wish to use is the same as that given in the program under the constant 'parallelop'.

When you start the program, you will first be prompted to enter the number of 8 bit input and output ports that you wish to use. The main menu will then appear; this will allow you to re-configure the number of input and output ports (1), output data to ports (2), input data from ports (3), and quit the program (4). Selecting output data will bring up another menu. This will give you the choice of entering data or loading the data that you have entered to the ports. If you choose to enter data, you are first prompted for the port to be used and then the data to be sent to the port. Once you have done that for all ports that you wish to program, select download and a brief delay will result whilst the data is loaded down to the ports and you will than be returned to the main menu.

Selecting input data from the main menu will produce a list of all the ports and the data that was present on them when you selected that option.

Testing the Unit

Having completed all of the boards that you need, you are now ready to test the unit. First, ensure that the units are connected together and that they are correctly connected to your computer's parallel port. Now apply a 5V power supply to connector 5, being careful to ensure that the polarity is correct. Next, run the control program on your computer. Output data to individual ports and use a logic probe or an LED connected to ground with a 330R resistor in series, to check that the data present at the outputs is the same as that which you entered. Now connect 5V to some of the inputs, run the input routine in the program, and check that the value being displayed for each port is the same as the data that you presented to the port. If there is a problem with any of the tests, the reason is likely to be due to bad solder joints, blown chips or an incorrect link to the computer. If this is the case, I suggest that you go back and check your work.

Possible Aplications.

A large number of aplications is possible, either to control external devices through relays etc or to monitor external conditions or events, for example temperature monitoring. The only restraint on the system that you wish to build up is the lowest speed at which you can safely operate.

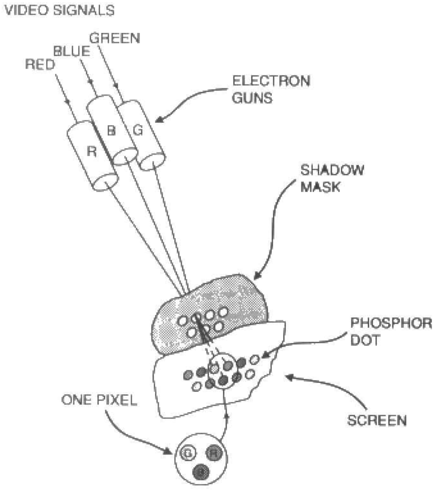
PC Clinic

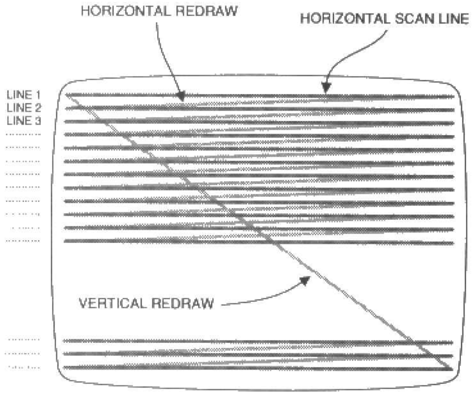
This month in PC Clinic, we look at the main PC output device, the video monitor, some of the faults which can be encountered and some of the ways in which they can be detected and cured

Although the actual electronics and construction of a standard CRT monitor are fairly complex, the theory behind its operation is not. An electron gun at the rear of the CRT tube fires an accelerated stream of electrons towards the front of the tube. When it hits the phosphor coating on the inside of the front of the tube, it causes that phosphor to glow.

The electron stream is very fine and is in fact focused to produce a very small glowing dot on the screen. An image is created on a raster scan system by moving or deflecting this dot very quickly across and down the screen so that it scans across the whole tube face. As it scans across the screen, the beam is turned on and off to create light and dark areas, this is the image that we see.

On a colour monitor the phosphor on the inner surface of the screen is arranged in a pattern of very small dots, arranged in groups of three, called triads; these can be seen in the accompanying diagram. The three dots in each triad are composed of different phosphors that produce different colours when hit by the electron beam. They are the three primary colours: red, blue, and green. It is the combination of these three colour dots in varying intensities into a single pixel which generates the wide range of possible screen colours.





In a colour monitor, there are actually three closely-coupled electron beams, one for each colour dot in the triad. Electron beams are deflected across the screen in a pattern similar to the second figure shown on this page. This is the raster scan pattern. It starts at the upper left corner of the screen and ends at the lower right corner. There is then a quick flyback as it moves back to the upper left corner to start another scan. On a modern monitor, the screen will be scanned between 50 and 90 times per second, a figure which is referred to as the 'screen refresh rate'.

The scanning is performed by electromagnetic coils which form a yoke around the neck of the CRT tube. One set of coils deflects the dot horizontally and the other deflects it vertically. To perform the scanning, these coils are fed with a carefully synchronised, ramped voltage that is derived from the monitor's timing circuitry. The same timing circuitry is used to determine when the electron beam is above a phosphor dot, thereby allowing the display circuitry to vary the intensity of the beam and therefore vary the intensity of the light produced by that specific phosphor dot.

It is these timing signals, their synchronisation, and voltage levels that we sometimes have to adjust in order to overcome problems such as: vertical and horizontal positioning, image size, pincushioning, and convergence. These controls are usually found at the rear of the monitor, sometimes actually within the case.

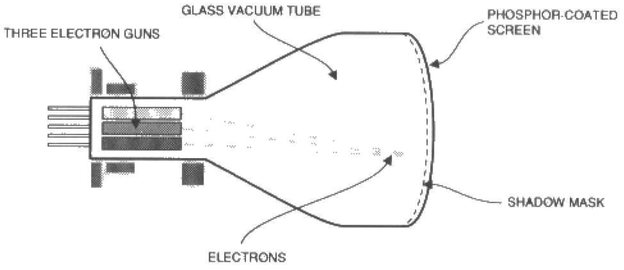
On early raster scan displays, the display resolution - the number of phosphor dots across the screen surface in both the horizontal and vertical directions - was sufficiently low that the electron beam could be focused to a fine enough point to hit just one dot at a time. However, as resolutions increased, particularly with colour displays, it became increasingly difficult to focus the beam on just one dot at a time with the result that the image lost its sharpness and its colour purity.

The solution to this problem lies in the use of a 'dot mask' or 'shadow mask', a very thin metal plate, pierced by a pattern of holes which exactly coincides with the phosphor dot triads on the screen. This effectively sharpens the image by blocking out stray electrons which might illuminate adjacent dots.

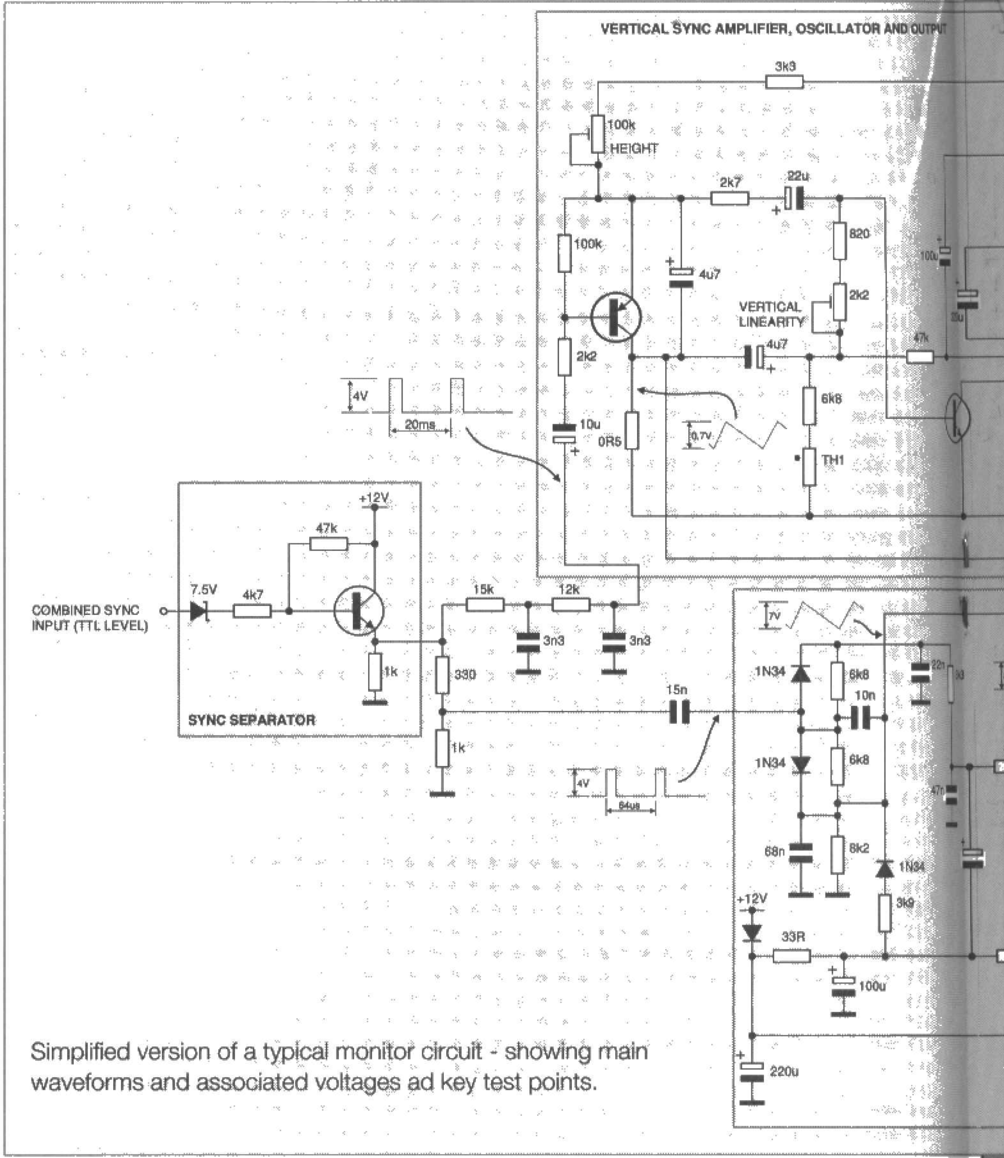
The mask and its underlying pattern of phosphor dots also determines the screen resolution; in other words the number of pixels. Most modern PCs will have monitors with screens having a resolution of between 800x600 and 1024x768 pixels. To a large degree the resolution is determined by the physical size of the CRT tube; a bigger tube will probably have a higher resolution.

Hence the reason why professional CAD systems usually have very large monitors is because the higher the resolution, the larger the image area that can be displayed. This means that a CAD system could well have a 21inch monitor with a resolution of 1024x768, whereas a word-processor could quite happily use a screen resolution of 640x480 on a 14inch screen. It should be noted that a screen resolution of 1024x768 has over 2.5 times as many pixels as one with a 640x480 resolution and can thus display 2.5 times as much data, or an image 2.5 times as large.

But resolution is also determined by the 'dot pitch', or the number of dot triads - in other words pixels - in a given distance of the screen surface. The closer the dots, the sharper the image. Dot pitch is usually measured in millimetres, and the smaller the number, the crisper the display. The current standard for graphics displays is .28mm, and for professional graphics displays .25mm; this latter pitch will give a very crisp clear display.



Fault testing in a typical computer monitor



Simplified version of a typical monitor circuit - showing main waveforms and associated voltages ad key test points.

For anyone with a reasonable knowledge of electronics and the appropriate test equipment ,it is not that difficult to locate and service faults in computer monitors. However, one does need the appropriate equipment. This consists of a multimeter with high voltage probe, and an oscilloscope. With these two instruments is should be possible to deal with most monitor faults.

The first procedure is, of course, to check that the fault actually lies in the monitor and not in the attached computer, the graphics display adapter, the connecting cable, or the power supply. Probably the easiest way to do this is simply to try another monitor which is known to work properly.

If the monitor is at fault then there are a number of fairly common faults which should be checked out first (here a copy of the manufacturer's service manual is essential). The following are some commonly encountered faults, they are as follows:

- Complete failure to work, no raster scan and all controls inoperative.

Possible Causes:

Power supply failure

check fuses, power supply AC and DC, supply rails, transformer, rectifiers, and regulators.

Horizontal output stage failure

check voltage supply at horizontal output stage

check waveform at collector of driver stage and horizontal output stage. If the former is normal while the latter is abnormal, replace output transistor

otherwise check continuity of flyback transformer windings

Horizontal driver or oscillator faulty

check signal at collector of driver stage; if abnormal, work back towards the oscillator stage

CRT dud

check heaters for continuity

check all DC voltages going into the CRT (use a high voltage probe and considerable care when testing the final anode supply). If any of the voltages are the same, disconnect power supply and check for shorts.

If these tests fail, replace CRT (take great care not to knock the tube as it could implode with considerable explosive force).

- Raster displayed but no video display

Possible Cause:

Video amplifier stage faulty

check DC supply to video amplifier

check output connector

check video waveforms at input and work towards output stage

- Data displayed but low brightness and display size increases and focus worsens as brightness turned up

Possible causes:

Horizontal output stage defective

check DC supply to horizontal output stage

check DC voltage to CRT electrodes(use a high-voltage probe and considerable care when testing the final anode supply)

Poor EHT regulation

check horizontal flyback transformer for short circuit

check EHT rectifier

- Data displayed but has poor contrast

Possible causes:

Video amplifier or output stage faulty

check DC supply on video amplifier stage

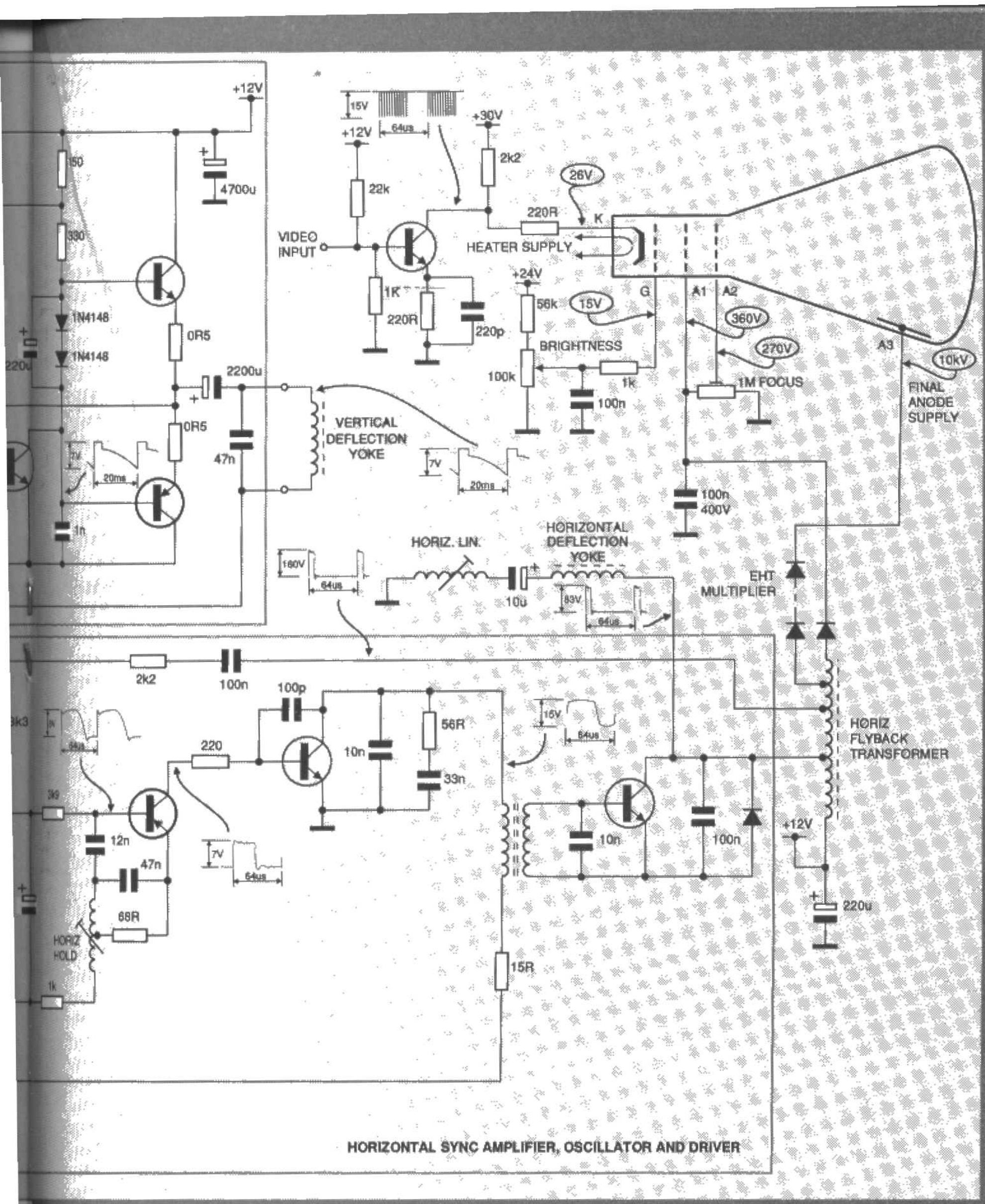
check video waveforms at input connector and work towards output stage

Contrast control

check pre-set potentiometer and adjust

WARNING:

Inside a computer monitor there are voltages which are potentially lethal. Unless capacitors are discharged, some of these voltages can still exist even when the machine is disconnected from mains power. Extreme care should be taken when touching or otherwise working on the circuitry whilst it is switched on, or even while it is switched off but still attached to the mains. If you have no experience of de aling with high-voltage equipment, and taking all the proper precautions, then it is strongly advised that maintenance of a monitor is left to someone who does have that experience.



Typical adjustment procedures

Very often an adjustment of the pre-set controls will need to be made after repairing a monitor or even after it has been moved or knocked. These pre-sets are, in most cases, simply a number of potentiometers and inductors, they are located in similar positions on most monitors and are usually well labelled. Accessing them will require removal of the case.

The first group of presets is the static convergence assembly, which consists of three thumbwheels or sets of levers mounted on the CRT neck. These, together with the blue lateral static convergence potentiometer, allow the three colour beams to be individually aligned. Any deflection distortion is usually corrected by the use of magnets on the CRT neck, although on more expensive systems they may be corrected by dynamic convergence circuits.

The next group of presets are the three colour adjustment potentiometers; one each for red, green and blue. These are usually located on a small PCB that is fitted onto the rear CRT connector. Focus is controlled by a potentiometer which is usually located as part of the line flyback transformer assembly, and horizontal is adjusted by means of a variable inductor on the main monitor PCB. Also on the main PCB are pre-sets controlling vertical and horizontal linearity and hold, plus controls on height, width, and horizontal oscillation frequency.

When making any adjustments to these pre-sets it is important to, first of all, observe proper safety procedures; there are potentially lethal voltages inside a monitor. Secondly, you will need to use some form of test software, such as CONVERGE, to generate a screen test pattern (it is a good idea to use a mirror to look at the image, thereby enabling one to safely adjust the display without having to lean around the monitor with all the potential danger that that could incur). Also, when making an adjustment, it is important to note the pre-set's original position and, if adjustment has no effect, return the preset to that position.

All the following adjustments relate to those necessary on a typical good quality colour monitor. They are organised as an adjustment sequence and this should be adhered to. The procedures are as follows:

1 Coarse raster adjustment - vertical and horizontal position

*horizontal sync - adjust H-hold to mid position, then adjust H oscillator inductor to middle of range that picture achieves sync.

*horizontal linearity - use H linearity inductor to adjust for equal width characters on left, right, and centre of screen.

*horizontal width - use width inductor to adjust for correct display width

*vertical sync - adjust V-hold potentiometer to centre of range that picture achieves sync.

*vertical linearity and height - adjust V-height to give display 70% of normal, then use V-line to give equal height characters at top middle and bottom of screen, then adjust V-hold to give full height display.

2 Static convergence

*using CONVERGE test pattern adjust for correct convergence at centre of screen, then adjust blue lateral preset until patterns align horizontally and vertically.

3 Colour purity

*display red pattern from CONVERGE and adjust preset potentiometer until all the screen appears in red, repeat for green and blue.

4 Repeat steps 1 to 3 as necessary

5 Dynamic convergence

*Using the preset potentiometers fitted to the convergence amplifier circuits, display CONVERGE convergence test pattern, then adjust for convergence first at screen centre, then at edges, then at four corners, repeat until convergence achieved.

6 Focus

*Adjust the dynamic focus preset to minimum, adjust focus preset to ensure screen is focused uniformly, then adjust dynamic focus for uniform focus at edges, repeat procedure as often as necessary. Use CONVERGE to aid procedure.

7 Colour balance

*Set R,G,and B presets to maximum, then reduce each in turn to achieve correct colour balance. Use the colour screens on CONVERGE.

8 Fine raster adjustment - vertical and horizontal

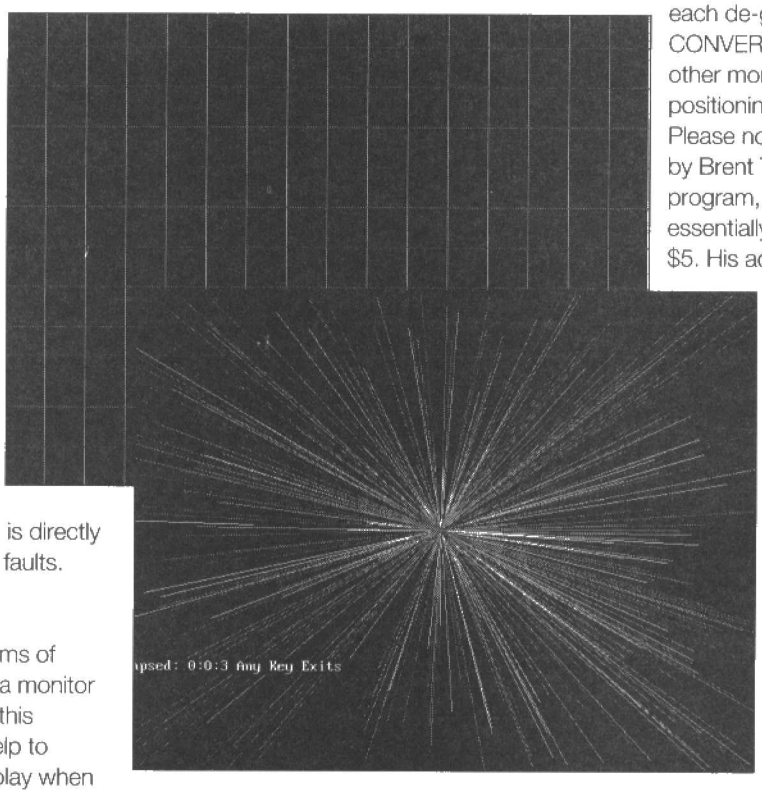
*Refine adjustments made in step 1 to counteract any changes brought about by subsequent steps.

9 Repeat steps 1 to 8 as necessary.

Diagnistics and adjustments

A lot of monitor problems are related to adjusting the various presets within the system; these control virtually every aspect of the actual display generation. Adjusted properly, they will produce a clear, bright display which has properly balanced colours and sharp readable characters; adjusted wrongly, they can lead to a display which is virtually unreadable.

Proper adjustment depends upon two factors. Firstly the use of diagnostic software aids to locate problems and make adjustment easier and, secondly, a knowledge of the adjustment procedures. This month's cover disk contains three pieces of diagnostic software, of which one - CONVERGE - is directly related to locating monitor faults.



```
psed: 0:0:3 Any Key Exits
```

Converge

There are five common forms of distortion encountered on a monitor which can be checked by this program, and which will help to produce a high quality display when adjusting the preset controls on a monitor. The forms of distortion which can be checked with CONVERGE are:

1. Pincushion - caused by faulty adjustment of horizontal deflection circuitry; test by holding a ruler against the vertical bars on the test grid. If the lines are not straight, then pincushioning could be a problem and adjustments might have to be made.
2. Keystone - This type of distortion is often caused by a malfunctioning power supply or weak flyback transformer; check by measuring the top and bottom lines of the grid. A difference of more than 1/8 inch indicates that keystoning is a problem. Repairing such a monitor requires component replacement and is probably best done by an authorised service facility.
3. Geometric - this can be a problem on cheap monitors and can be checked by seeing whether the corners of the displayed grid are at 90degrees. If not, then geometric distortion due to misplaced magnets on the neck of the picture tube could be the cause. This should be checked for when buying a new monitor.
4. Colour convergence - the grid can be used to check that the three electron guns on a colour monitor are accurately aligned with the phosphor dot triads. If they are not, then the display will look out of focus or show a slight shadow. For proper alignment, the vertical lines are yellow which requires a precise alignment of red and green guns; the horizontal lines are magenta, which requires a precise alignment of the red and blue guns, while the outer rectangle of the grid is white which requires an exact alignment of all three electron guns. If there is a misconvergence then the colours of the lines will separate or display colour-shadowing.
5. Testing colour - by repeatedly pressing the space bar, CONVERGE will display a white screen, and screens in each of the three primary colours, red, green, and blue. When buying a new monitor, it is a good idea to run this test and see if the colours are pure and intense. Also check that the colour is even across the display; there should be no blotchy areas, shadows or paling of colour in the corners. If the screen does show blotchiness, then de-gaussing may solve it (this is de-magnetising the shadow mask). Some monitors have a de-gaussing button, others do it every time the monitor is switched on. Try de-gaussing two or three times, allowing several minutes between each de-gaussing cycle.

CONVERGE can, of course, be used to aid other monitor adjustments, such as image positioning, height adjustment, etc. Please note that CONVERGE was written by Brent Turner and is a 'Donate-ware' program, a form of Shareware; this means essentially that if you like it then send him $5. His address is: Brent Turner, P.O.Box 3612, Fullerton, CA 92634-3612.

Burnin

This program is designed to exercise every part of a PC's system, from monitor, to printer, to disk drives and processor. In so doing it is designed to locate any area where faults are likely to occur.

It is designed to be run continuously for anything up to 72 hours and, in so doing, will simulate the use of the system for several months. This intensive work out of the system should show up any latent faults.

It is a good idea to run Burnin for 72 hours when you first buy a system, after it has been repaired, after new hardware - in particular adapter boards - has been added, and of course if an intermittent fault is suspected. If there are problems the Log file will pinpoint them.

For full details on using this software see the extensive details contained in the text file on the disk. Please note that this is a Shareware program, copyright George Campbell 1992, and is not in public domain. So if you like it and want to use it then please become a registered user, Registration costs just $15 and a form is on the disk.

3D Bench

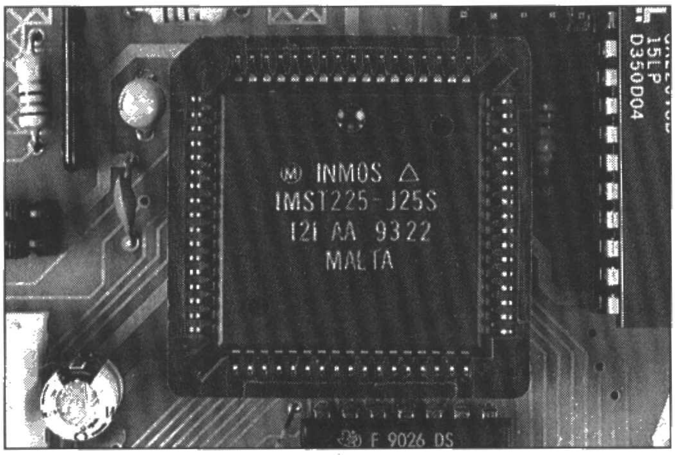
This is a simple benchmark program which allows the user to compare the display performance of PC-compatible VGA systems. This is an invaluable help if you are thinking of buying a new system or upgrading an old system which will be used for graphics intensive/multimedia type applications. It generates a moving three-dimensional display and then checks how fast the display was generated to produce a figure for the average number of screen updates achieved in one second.

This can vary from two to three frames per second on an old 12MHz AT with an 8bit VGA card to 50fps on a 66MHz 486DX2 with VESA bus graphics card. Obviously the higher the figure, the better suited the system is to graphics-intensive applications. This is a public domain program from New Dimension International Ltd.

Next...
In the next issue we will be looking at the hardware which generates the display shown on the monitor, at graphics display cards, and at some of the other display technologies which are in use today.

Transputer

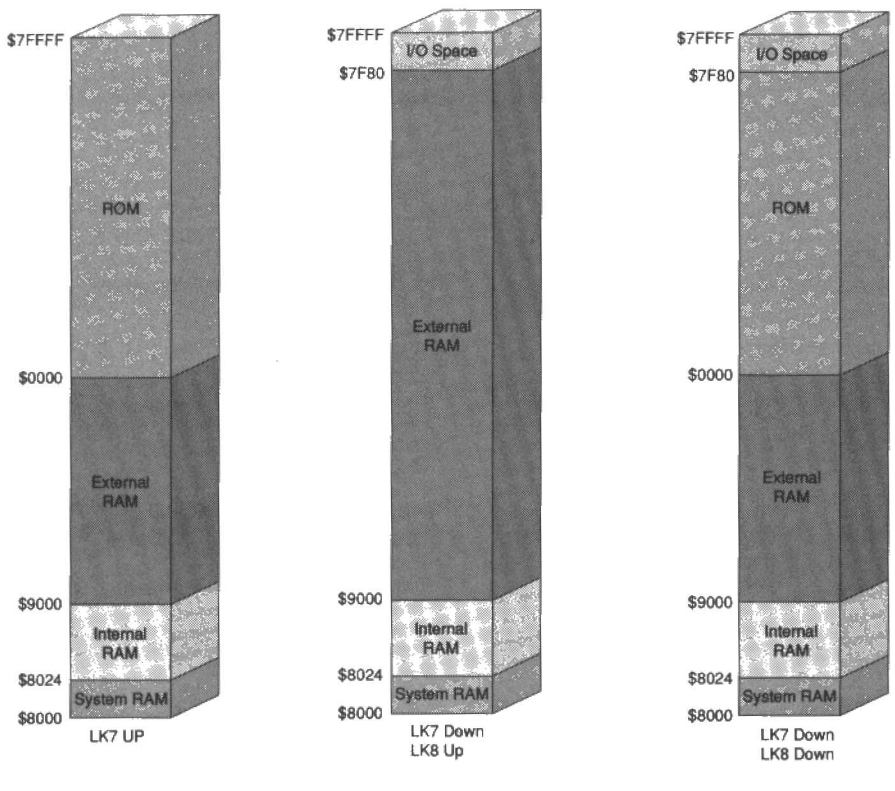
Based Single Board Computer



Project designers Andy Papageorgiou and Mark Robinson look at testing the Transputer board developed in last month's issue of ETI.

Last month's article described the construction and initial testing of the T225 card. This month we describe the construction of the Link Adaptor and use it to run a simple test program on the transputer.

Fig.3. The three possible system memory maps



The Link Adapter Circuit.

The Link Adapter circuit is designed to allow IBM compatible PCs fitted with a parallel I/O card to communicate with the transputer via a link. Although it is not essential to build the link adaptor to use the transputer card, it allows a PC to be used to boot the transputer, to interrogate its status, and to receive data for graphical display etc. The card was designed for use with the Maplin 8255 based PC I/O card although it should work with other manufacturer's 8255 cards although the pinout of their connector may be different.

The circuit diagram of the link adaptor is shown in figure 1. The method of construction is left to the constructor; we built it using point-to-point wiring on the end of a piece of 100x160mm stripboard, the remainder of the area was used as a simple backplane to allow a number of cards to be connected together.

Most of the work is done by IC1, an Inmos link chip designed for this purpose. The clock signal for IC1 is generated by IC2b and associated components. For correct operation the frequency of this clock must match the one on the transputer card to within 400ppm. Although this stability is easily achieved with low cost crystals, care must be taken when laying out the circuit to ensure that stray capacitance does not detune the oscillator. Mount all the clock components close to IC2 and cut all the tracks as close as possible to the IC. The PLL capacitor C1 must also be mounted as close as possible to IC1.

The pinout of the transputer card edge connector (viewed looking into the pins) is shown in figure 2. A backplane can be made simply by mounting a number of DIN41612 sockets onto a piece of stripboard with track cuts between the two rows of pins. In such a system the links are connected as two chains running the length of the backplane, links 0 and 2 connect to links 1 and 3 of the next card in the rack. Link 3 of the first card is connected to the adaptor circuit, and the two links left on the last card can either be taken to an expansion ceonnector or doubled back into the first card. The local reset and local analyse lines are also propagated down the chain. There are also a number of uncommitted pins which can be used to propagate control signals down a back-plane.

The chip supports both the standard 10Mbit/s link speed and the extended 20Mbit/s link speed, selected using LK1. Pin 15 can be directly connected to +5V to permanently select the faster speed since there is no real advantage to using the slow speed, except for compatibility with now obsolete transputer types.

Option Jumper Settings.

Before the transputer can run a program, it is necessary to set up a number of jumper options on the board, mainly concerned with the ROM. the functions of the jumpers are as follows:

LK1 (ResetAction).

When the card is used with a backplane, there are two reset signals: Global Reset, which is connected to every card and acts as a complete system reset, and Local Reset. If LK1 is down ,the Local Reset is simply passed to the next board in the chain; if it is up, the Local Reset to the next board is controlled by bit 6 of the data direction register (IC10). This allows boards further down the backplane to be reset under software control. When the card is not used in a backplane, this jumper can be left unmade.

LK2 (AnalyseAction).

This jumper is the same as the ResetAction jumper, except that it controls the Local Analyse signal on the backplane.

LK3 - LK5 (LinkSpeed).

These jumpers control the speed of the links, according to the table given below

LK3	LK4	LK5	speed
made	made	made	20Mbit/s
made	made	open	Link0 20MBit/s Links1-3 10Mbit/s
made	open	made	Link0 10MBit/s Links1-3 20Mbit/s
made	open	open	10Mbit/s
open	made	made	10Mbit/s
open	made	open	Link0 10MBit/s Links1-3 5Mbit/s
open	open	made	Link0 5Mbit/s Links1-3 10Mbit/s
open	open	open	5Mbit/s

LK6 (BootFromROM).

When this jumper is made, the transputer will boot from the ROM; otherwise it will wait for a boot message on any link.

LK7 (PageAtReset).

If this jumper is down, the ROM will be paged in at reset; otherwise it will be paged out.

LK8 (PageAction).

This jumper determines how the ROM page mechanism works. Figure 3 shows the three memory configurations of the system. If LK8 is up, paging the ROM toggles the memory map between figure 3a and figure 3b. If LK8 is down, paging the ROM toggles between figure 3a and figure 3c. The memory map in figure 3c allows a minimum system to be built without any external RAM. In this case, the transputer would page the ROM out after booting to allow access to the I/O space, but the remainder of the ROM would still be visible. This mode has little value if the external RAM is fitted since it prevents any access to the upper half of the RAM.

LK9 (ROMSpeed).

If this jumper is down, the GAL will request three wait states when accessing the ROM; if it is up, the GAL will request six wait states. Three wait states are sufficient if the ROM has an access time of 150ns or faster.

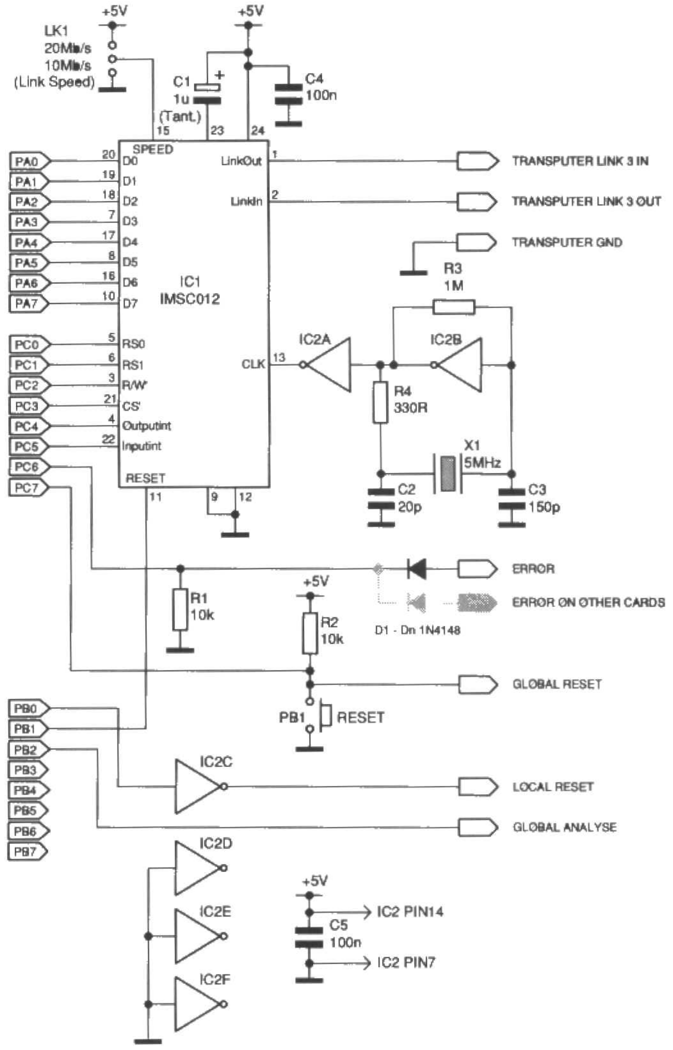


Fig.1.The link adaptor circuit

LK10 (WritableROM).

If this jumper is down, it signals that the ROM is a writable device (EEPROM or NVRAM). If an EPROM is fitted, this jumper MUST be set to up otherwise the PGM' pin of the EPROM may be asserted with unpredictable results. When the link is up, writes to the ROM will be redirected to the corresponding RAM location, and hence a program can be copied from ROM to RAM simply by reading the memory and writing the data back to the same locations.

LK11 and LK12 (ROMPinout).

These jumpers cater for the slight difference in pinouts between the various ROM devices that can be used with the card. Set the jumpers according to device type as follows (treat NVRAM as an equivalent sized EEPROM):

	LK11	LK12
2764, 27128	up	down
27256	up	up
28256	down	down

Listing 1. Program code for TBOOT.C

```
    #include <stdio.h>
    #include <dos.h>
    #include <time.h>
    #define BASEADD 0x300                    /* Base address
of the 8255 card */
    #define PORTA (BASEADD + 0)
    #define PORTB (BASEADD + 1)
    #define PORTC (BASEADD + 2)
    #define CONTROL (BASEADD + 3)
    #define MAKEINPUT 152
    #define MAKEOUTPUT 136
    #define TIME_OUT 5                 /* wait 5 seconds before
timing out */
    #define CS 8
    #define RW 4
    #define OPINT 16
    /* Wait for port C bit 4 to become active, indicates that the
link
      is ready to accept a byte of data */
    void wait_for_link()
    {
    time_t start;
    start = time(NULL);
    while (!(inportb(PORTC) & OPINT))
            if ((time(NULL)-start) > TIME_OUT)
                    {
                    printf(“Link timed out\n”);
                    exit(-1);
                    }
    }
    /* Write the byte contained in val to the link adapter register
reg */
    void write_register(reg, val)
    int reg;
    unsigned char val;
    {
    /* Put the required byte on port A */
    outportb(PORTA, val);
    /* Pulse the link adapter CS line low to write the byte */
    outportb(PORTC, (unsigned char)(reg+CS));
    outportb(PORTC, (unsigned char)(reg));
    outportb(PORTC, (unsigned char)(reg+CS+RW));
    }
    unsigned char read_register(reg)
    int reg;
    {
    unsigned char temp;
    outportb(CONTROL, MAKEINPUT);
    outportb(PORTC, (unsigned char)(reg+RW+CS));
    outportb(PORTC, (unsigned char)(reg+RW));
    temp = inportb(PORTA);
    outportb(PORTC, (unsigned char)(reg+RW+CS));
    return(temp);
    }
    /* Initialise the link adapter status registers */
    void setup_link_registers()
    {
    unsigned long i;
    outportb(PORTC, CS+RW);
    outportb(PORTB, 3);
    for(i=0; i<100000; i++);
    outportb(PORTB, 0);
    for(i=0; i<100000; i++);
    write_register(2, (unsigned char)3);
    write_register(3, (unsigned char)3);
    }
    void main(argc, argv)
    int argc;
    char **argv;
    {
    FILE *hexfile;
    int i, j, fbit;
    int num_bytes, num_rows, last_row, this_row;
    unsigned int val;
    /* Make port A an output, since all link transfers are writes
*/
    outportb(CONTROL, MAKEOUTPUT);
    setup_link_registers();

    if (argc!=2)
            {
            printf(“Usage: %s <hex file>\n”,argv[0]);
            exit(-1);
            }
    if (!(hexfile = fopen(argv[1],”rt”)))
            {
            printf(“Can’t find file %s\n”,argv[1]);
            exit(-1);
            }

    /* Count the number of entries in the file, including
preamble */
    num_bytes = 0;
    do
            {
            fscanf(hexfile,”%X”,&fbit);
            num_bytes++;
            } while (!feof(hexfile));
    rewind(hexfile);
    num_rows = (int)(num_bytes/17);
    last_row = num_bytes%17;
    num_bytes = 16*num_rows;
    if (last_row!=0) num_bytes+=last_row-1;

    wait_for_link();
    write_register(1,(unsigned char)(num_bytes));
    printf(“Number of bytes = %d\n”,num_bytes);

    /* Hex file consists of an address followed by 16 bytes on
each row.
    A minor complication is that the last row may have less
than 16 bytes */
    this_row = 16;
    for (i=0; i<num_rows+1; i++)
            {
            if (i==num_rows) this_row = last_row-1;
            if (this_row != 0) fscanf(hexfile,”%X”,&fbit);
            for (j=0; j<this_row; j++)
                    {
                    fscanf(hexfile,”%X”,&val);
                    printf(“%.2X “,val); fflush(stdout);
                    wait_for_link();
                    write_register(1,(unsigned char)val);
                    }
            printf(“\n”);
            }
    fclose(hexfile);
```

Listing 2. Program code for T225B.ASM

```
        ; Error Light flasher and port exerciser.
        ; (c) 1994 Mark Robinson
        ; Flashes error light and one of the O/P ports

port EQU 0              ; Which port is exercised

        AJW 5           ; Adjust workspace pointer
        CLRHALTERR      ; Clear "Halt on Error"
        TESTERR                 ; Clear error flag

        MINT            ; Minimum integer
        STLF            ; Initialise queue pointers so that
        MINT            ; multi-processing may be used
        STHF            ;

        LDC 0           ; Initialise loop control
        STL 0
        LDC 11
        STL 1
lLp:    MINT            ; Null value
        LDL 0           ; Index
        MINT            ; Base address of transputer
        WSUB            ; Calculate "channel" address
        STNL 0          ; and store null value in it

        LDLP 0          ; Point at loop control block
        LDC lLpEnd-lLp  ; Offset to start of loop
        LEND            ; Repeat

lLpEnd: LDC 0           ; Initialise and start the clocks
        STTIMER                 ; which enables times-
licing

        LDC $3F
        LDC $7FB8
        STNL 0          ; Set all ports to output

        LDC 3500
        STL 4           ; Error light ON time
        LDC 4500
        STL 6           ; Error light OFF time

LLabel: SETERR          ; Turn on error light
        LDC 0
        LDC $7F80
        STNL port*4     ; Set port to 00
        LDTIMER
        LDL 4
        SUM
        TIN             ; Wait
        TESTERR                 ; Turn off error light
        LDC $FF
        LDC $7F80
        STNL port*4     ; set port to FF
        LDTIMER
        LDL 6
        SUM
        TIN             ; Wait some more
        J LLabel ; Do it all again
```

Running a Program.

Now that the jumpers are set to sensible places, it is finally time to run our first program on the transputer. First type in listing 1 and save it as 'tboot.c' (alternatively, the program is on the example disk to save wear on the fingers). The program was written in Turbo C but it should compile with Microsoft C. Set the '#define BASEADD 0x300' line to the base address of your I/O card, and compile the program with the command (for Turbo C)

```
tcc tboot.c
```

which will generate the executable 'tboot.exe'. If you don't have a C compiler, the executable is on the example disk. This program will be used to send boot programs down the link to the transputer.

The assembly language listing for the first test program is given in listing 2. To assemble this program you will need the assembler on the examples disk, so there's no point typing it in. If you don't have the disk, type in the hex listing given in listing 3 and save it as 't225b.hex'. If you have the disk, assemble the code using the command tasm t225b.asm /h after ensuring that the tasm.exe program is copied into a directory that is in your path. The file tasm.doc contains more details about the assembler.

Ensure that both the transputer and link adaptor are reset and boot the transputer using the command

```
tboot t225b.hex
```

As each byte is sent to the link, its hex value is printed on the screen and when the entire program is sent, the error LED should flash with a period of around 1 second. The project has now broken two records; it is the most expensive LED flasher ever published in a hobbyist magazine, and it is the project which can have most ICs removed before malfunctioning. In fact, all the ICs except the transputer and IC6 could be removed without stopping the LED flashing.

There are two ways that the program could fail to work. The least likely is that the program is sent to the transputer without a problem but the light doesn't flash. Assuming that the listing has been typed in correctly, only the components around the transputer error pin can be at fault; check that Q1 is OK and that the LED is mounted the correct way round.

The most likely failure mode is that the boot program will print the message "Link Timed Out". If this occurs intermittently or part way through booting the code, the most likely cause is timing problems. Check that the clock is clean and has sharp transitions. If you have a frequency counter check that the

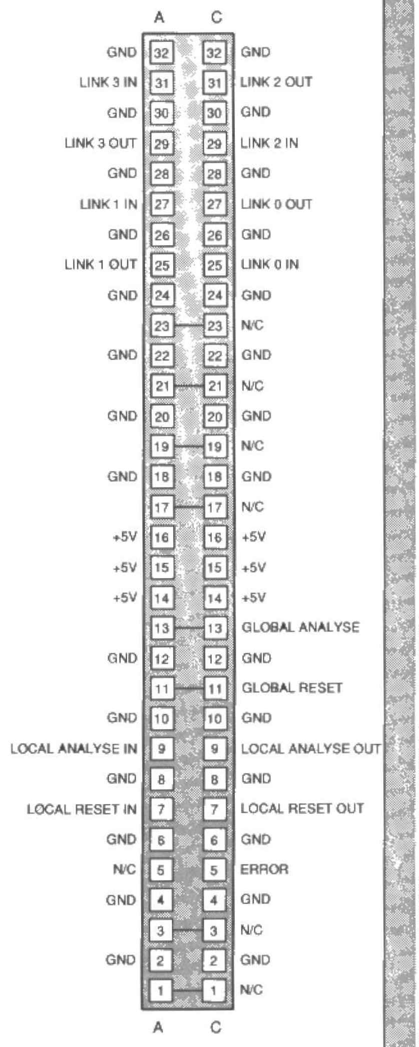


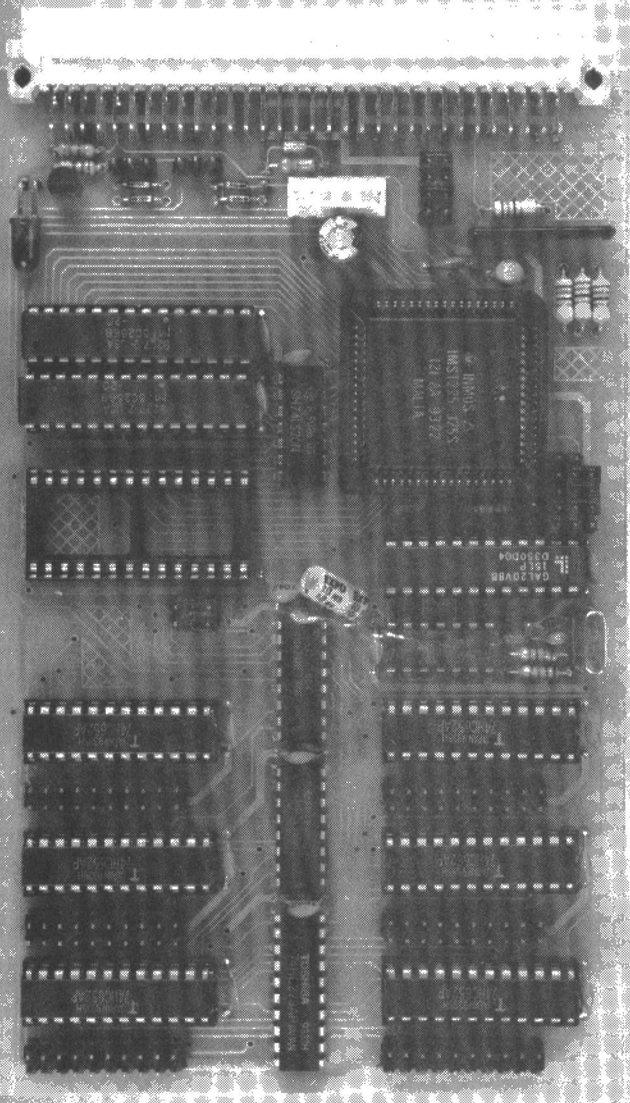
Fig.2. Pinout of the edge

Listing 3. Program Code for TEST.C

```
#include <stdio.h>
#include <dos.h>
#include <time.h>
#define BASEADD 0x300                  /* Base address of the
8255 card */
#define PORTA (BASEADD + 0)
#define PORTB (BASEADD + 1)
#define PORTC (BASEADD + 2)
#define CONTROL (BASEADD + 3)
#define MAKEINPUT 154
#define MAKEOUTPUT 138
#define TIME_OUT 1                     /* wait 2 seconds before
timing out */
/* Wait for port C bit 4 to become active, indicates that the link
  is ready to accept a byte of data */
void wait_for_link()
{
time_t start;
start = time(NULL);
while (!(inportb(PORTC) & 16))
        if ((time(NULL)-start) > TIME_OUT)
                {
                printf("Link timed out — please reset the
transputer\n");
                exit(-1);
                }
}
/* Write the byte contained in val to the link adapter register
reg */
void write_register(reg, val)
int reg;
unsigned char val;
{
/* Put the required byte on port A */
outportb(PORTA, val);
/* Pulse the link adapter CS line low to write the byte */
outportb(PORTC, (unsigned char)(reg+8));
outportb(PORTC, (unsigned char)(reg));
outportb(PORTC, (unsigned char)(reg+8));
}
unsigned char read_register(reg)
int reg;
{
unsigned char temp;
time_t start;
start = time(NULL);
while (!(inportb(PORTC) & 32))
        if ((time(NULL)-start) <
 TIME_OUT)
                {
                printf("Link read timed out\n");
                exit(-1);
                }
/* Pulse the CS line while holding R/W high */
outportb(PORTC, (unsigned char)(reg+4+8));
outportb(PORTC, (unsigned char)(reg+4));
temp = inportb(PORTA);
outportb(PORTC, (unsigned char)(reg+4+8));
return(temp);
}
/* Initialise the link adapter status registers */
void setup_link_registers()
{
outportb(PORTC, (unsigned char)8);
write_register(2, (unsigned char)3);
write_register(3, (unsigned char)3);
}
void poke_link(add, val)
unsigned short add,val;
{
outportb(CONTROL, MAKEOUTPUT);
wait_for_link();
write_register(1, (unsigned char)0);
        /* send control byte 0 — poke */
wait_for_link();
write_register(1, (unsigned char)(add%256));
wait_for_link();
write_register(1, (unsigned char)(add/256));
        /* poke address low byte first */
wait_for_link();
write_register(1, (unsigned char)(val%256));
wait_for_link();
write_register(1, (unsigned char)(val/256));
        /* poke data low byte first */
}
unsigned short peek_link(addr)
unsigned short addr;
{
unsigned char lo,hi;
wait_for_link();
write_register(1, (unsigned char)1);
        /* send control byte 1 — peek */
wait_for_link();
write_register(1, (unsigned char)(addr%256));
wait_for_link();
write_register(1, (unsigned char)(addr/256));
        /* peek address low byte first */
outportb(CONTROL, MAKEINPUT);
lo = read_register(0);
hi = read_register(0);
        /* get return value */
return((unsigned short)(lo+hi*256));
}
void main()
{
unsigned short addr, data, ret;
setup_link_registers();
while (1) /* do this forever */
        {
        printf("Address (hex) ?\n");
        scanf("%x",&addr);
        printf("Data (hex) ?\n");
        scanf("%x",&data); /* get an address and value */
        poke_link(addr, data); /* Poke the transputer's
memory */
        ret = peek_link(addr); /* Peek the transputer's
memory */
        printf("Address %XH returns
%XH\n",(int)addr,(int)ret);
        }
}
```

frequencies of the two clocks are within 400ppm and stable. If not, attempt to eliminate all sources of stray capacitance - long component leads or long track lengths for example. As an absolute last resort, CX can be replaced by a 10pF capacitor and 2-10pF trimmer, adjusted for reliable operation.

If the timeout message always appears before the first byte of the listing is printed on the screen, then the connection between the host computer and the link adaptor is suspect. Check that your I/O card works, and that the base address corresponds to that which you defined in the source code. The tboot program uses the signal on pin 4 of the link adaptor chip to determine if the link is ready to accept data; if the link times



out even when this pin is high then the signal is not getting to the computer. Before waiting for the link, the program sends a short burst of initialisation data to the adaptor chip. Use a logic probe to check for pulses on the CS' pin, the R/W' pin and the RS0 and RS1 pins before the program times out.

If the timeout message appears immediately after the first byte has been sent, then the connection between the link adaptor and transputer is suspect. Check that the link speed setting on the link adaptor agrees with the transputer setting and that the clock is reaching the adaptor chip. Check that the links are wired correctly - link out to link in. Use a logic probe to check for a burst of activity on the adaptor's link out pin and watch for the transputer pulsing the link in pin to acknowledge receipt of the byte. If the acknowledge pulse isn't sent, check the connections to the transputer carefully, and check the option jumpers to ensure that the transputer isn't attempting to boot code from ROM.

Although the flashing error light confirms that data is being sent to the board properly, it does not test the board fully. However, the error light flashing program also pulses the lines of Port 0. Check that the lines are pulsing using a logic probe, 'scope or an LED. If they are, then the card is almost certainly fully functional. If not, check for activity around the I/O decode circuitry.

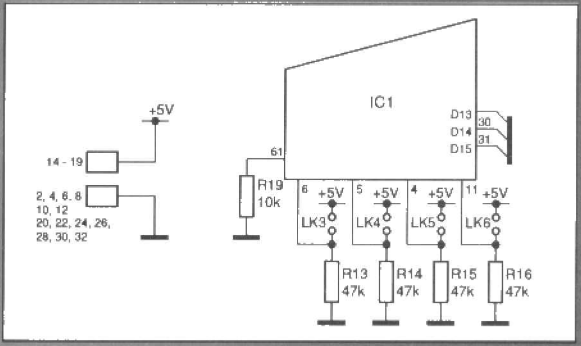
The program in listing 2 is a useful test program for use in debugging the transputer card. A feature of the transputer is that while waiting for a boot program from link, special peek and poke instructions can be sent to modify and examine the transputer memory. The program in listing 2 exploits this feature. When run it will prompt for an address and data word. It will then poke the data into the address and read it back. Use this program in conjunction with the memory maps in figure 3 to satisfy yourself that the external RAM and ROM are functioning, and that the ROM is paged out when it should be. Note the unusual twos complement addressing convention; the lowest address is 8000H, increasing through 0000H to 7FFFH. The first 26H bytes of memory are used by the system and should not be written to. To test the I/O, first write FFFFH to address 7FB8H to set the ports to output, the six ports then appear at addresses 7F80H, 7F88H, 7F90H, 7F98H, 7FA0H and 7FA8H.

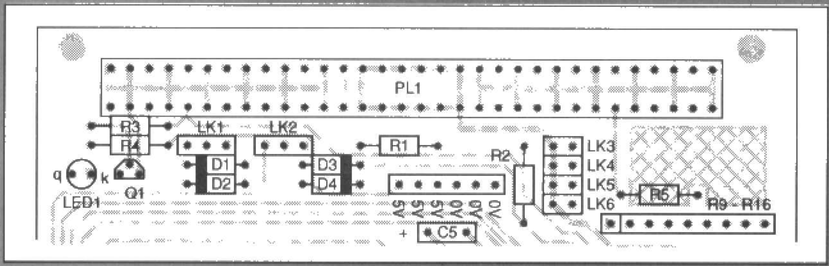
NEXT MONTH

Next month we will look at programming the card, describe the I/O region of the memory map in detail and provide a bootstrapping program to help with concurrent program development.

Correction of errors in Part 1 of Transputer project published in October 1994 issue of ETI.

Unfortunately, due to software incompatibility, parts of some diagrams did not appear properly. We are therefore reprinting these parts and apologise for any inconvenience caused to readers.

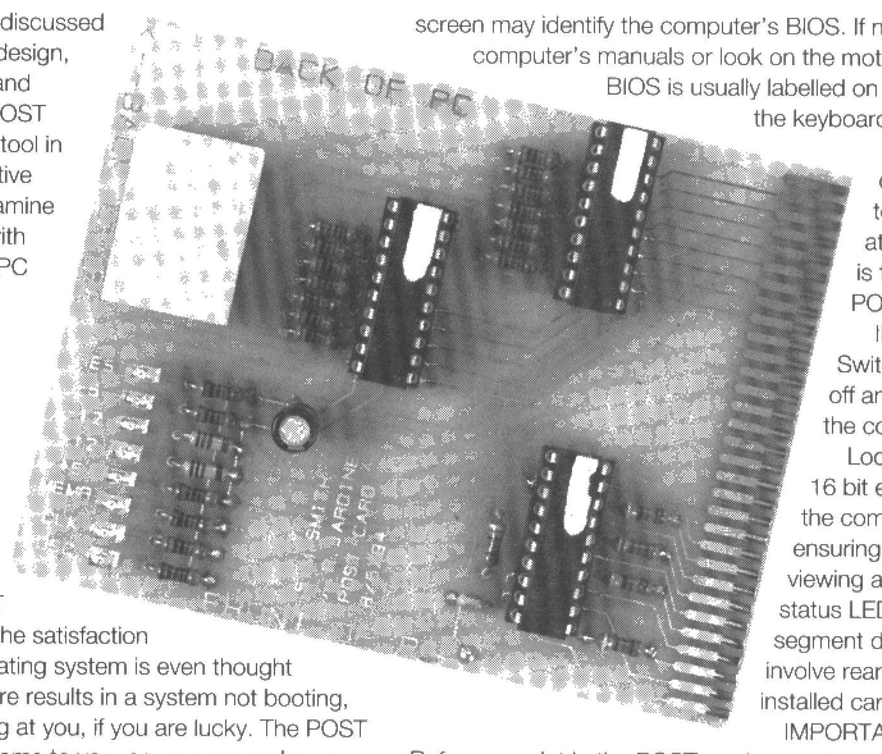




PROJECT

The ETI POST CARD

In part 2 of their PC Power On Self Test card project, Stephen Smith and shows A.Jardine how the card can be used to locate faults in a system



Last month we discussed the concept, design, construction and testing of the POST card. An invaluable tool in the fight against uncooperative PCs. This month we will examine the use of the POST card with three of the most common PC BIOSs (AMI, Award and Phoenix).

Just to recap on POST. Power On Self Test is the PC's defence against the ravages of time (and dumb users). It tests the main system components to ensure they are in a fit state to perform the task asked of them. All the POST tests have to be passed to the satisfaction of the BIOS before the operating system is even thought of. This means that any failure results in a system not booting, just sitting there and beeping at you, if you are lucky. The POST card relates the PC's symptoms to you, so you can make your diagnosis.

The POST card does, however, have another use. This is as a tool in software development and maintenance. How many times have we put a small print statement in to monitor the progress of the program and see where it hits that infinite loop. The problem with this method is that, in real time programs, the time taken in processing that print statement can seriously effect the program's operation. If the POST card is used as a monitor, the complex print statement, with its disrupting effect upon the screen, is replaced by a simple output command. Listing 4 shows how a simple loop can be monitored with very little software overhead. Even the command variable can be displayed. Programs that are going into the field cannot display such diagnostic messages on the screen, because they would add an unnecessary distraction to the user. But you can send them to the POST card with no noticeable processing overheads.

User Instructions.

Before you even consider using the POST card, power up your PC. Note exactly what the computer does. Messages on the screen may identify the computer's BIOS. If not, refer to your computer's manuals or look on the motherboard, (as the BIOS is usually labelled on an EPROM and the keyboard controller).

If the computer does not attempt to boot the operating system, then it is time to use the POST card.

IMPORTANT - Switch the computer off and carefully remove the computer case.

Locate a spare 8 or 16 bit expansion slot in the computer to be tested, ensuring that there is easy viewing access for the status LEDs and seven-segment display. (This may involve rearranging already installed cards if necessary.)

IMPORTANT WARNING - Before you slot in the POST card, note that the card is marked with directions. These tell you where the card should be, in respect to the back and front of the PC. It is vital that the card is not plugged in the wrong way around, as damage to the card and PC may result. Not a good start to repairing your PC.

Turn on the power to the PC.

If all is well the status LEDs (on the left of the board), should be lit up.

Problems indicated by status LEDs.

If all the status LEDs are functioning correctly then the next step is to analyse the POST codes. (Please note that the memory read indicator, MEMR, may be dark or flicker if memory access upon the bus is infrequent.)

If the power supply LEDs do not light, check that the power is getting to the machine, the mains input cable and the power connector to the motherboard. If the problem persists, repair or replace the power supply unit.

One use of the power supply LEDs that we found was in debugging a supposedly faulty COM Port. A machine's mouse had died and it was traced to the COM Port to which it was connected being totally dead. The multi-IO card was replaced only to find the problem continued. When the POST card was inserted in the machine, the ±12V supply LEDs did not light. This was traced to the power lines on the motherboard having been eaten away by a leaking NiCad battery.

If RES LED does not light then the system is in a continuous reset state. The reset LED should be off for about half a second after power up and whenever the reset button is pressed. Check the reset circuitry and power supply, as the action of the reset circuit depends upon these.

If OSC LED does not light, check the bus oscillator circuitry and 14.318MHz crystal. This signal is not used by the majority of cards, but is often used by the some hard disk controllers and video cards. The symptoms are inability to read drives and complete loss of synchronisation on the monitor.

If CLK LED does not light, check the main CPU clock, CPU and bus interface circuitry.

If MEMR LED does not light then the microprocessor is not accessing the bus or not running. Check the processor, bus interface circuitry and co-processor jumper settings.

Problems indicated by POST.

If the system is capable it might beep at you. Its swan song maybe, but an invaluable way of find out what is wrong.

Audio error codes.

The PC beeps once every time the POST sequence is completed successfully. When something goes wrong, however, it may become more talkative. Table 2 lists the audio error codes for the AMI BIOS. As you can see, the AMI has a limited vocabulary, talking about only the most important things. The phoenix is a totally different beast, being a positive chatter box when allowed, see table 3. Having such a large vocabulary lets it describe exactly where it hurts, even down to which bit in the RAM failed the testing. If the PC is not booting and has been struck dumb, the POST cards seven-segment displays are now the place to look

POST codes.

The seven-segment displays on the POST card act as a window to the POST testing. If the test unexpectedly stops, the code displayed directs you to the source of the problem. If no fault is detected, the PC boots and the final POST code tells you this. Tables 3, 4 and 5 list the POST codes for Phoenix, Award and the most common, AMI BIOS.

The following are POST codes for AMI Colour BIOS and hints where to look for faults.

06 BIOS Checksum. The BIOS ROM's is checksummed to validate its operation. The failure of this indicated that the EPROM is not being read correctly. The first thing to do is ensure that the EPROM is seated properly in its socket. Address lines stuck or an aged EPROM could also give this error.

13 Chipset initialisation/auto Memory detect. A memory or memory controller fault is indicated here. Try re-inserting your memory (the SIMMs usually). This is one of the most common faults on PCs.
2CHanding control to video ROM. The BIOS passes control to the VGA BIOS to allow it to initialise the display. A fault on the VGA card is most likely so replace it and retest. An address conflict with another card is possible. This is another common fault.

4F or 51 Below or above 1MB memory failure. A memory or memory controller fault again.

8F Floppy disk set-up. A floppy controller fault. Make sure that the floppy drive connection and power cables are secure.

97 Handing control to optional ROM. Fault with intelligent hard disk controller (i.e. any hard disk controller with its own BIOS) or network card.

A1 Cache memory fault. Consider this, the PC hangs at the start-up screen with not a sound. This could be fun if you have not got your POST card to hand. Try disabling cache in the set-up first, if machine now OK, find faulty RAM.

00 If all tests have run and machine does not boot, suspect software fault or faulty disk drive or disk controller or DMA chip.

The battery backed CMOS RAM, timers and DMAs are frequently in a single chip (the 82C206) which has a high mortality rate and can easily be replaced if it is socketed.

Control of the A-20 line is in the hands of the keyboard controller and is used to play about with the high memory. The keyboard controller (usually a 8042) is commonly socketed and can easily be exchanged.

One general piece of advice I can give is that removing all expansion cards can help track down the problem, even if the problem has no direct connection with the expansion cards. For instance, a memory conflict caused by defective address decoding on a card. If the POST tests stop at an interrupt or DMA test, we have found that this may be caused by a rogue expansion card. Again, removing all cards to prove their innocence in a retest is advised. This form of problem is unlikely to occur in a previously working system as it highlights subtle incompatibilities in cards.

All in all, fixing PCs can be infuriating but very rewarding and ,most of all, fun. Please feel free to contact us (via ETI) about your experiences. Have fun!

Table 2

AMI POST Beep Codes

Fatal Errors.

1 short beep	DRAM refresh failure.
2 short beeps	Parity circuit failure.
3 short beeps	Base 64K RAM failure.
4 short beeps	System timer failure.
5 short beeps	Processor failure.
6 short beeps	Keyboard controller GAte A-20 error.
7 short beeps	Virtual mode exception error.
8 short beeps	Video memory R/W test failure.
9 short beeps	ROM BIOS checksum failure.

Non-fatal Errors.

1 long, 3 short beeps	Conventional/extended memory failure.
1 long, 8 short beeps	Display/retrace test failure.

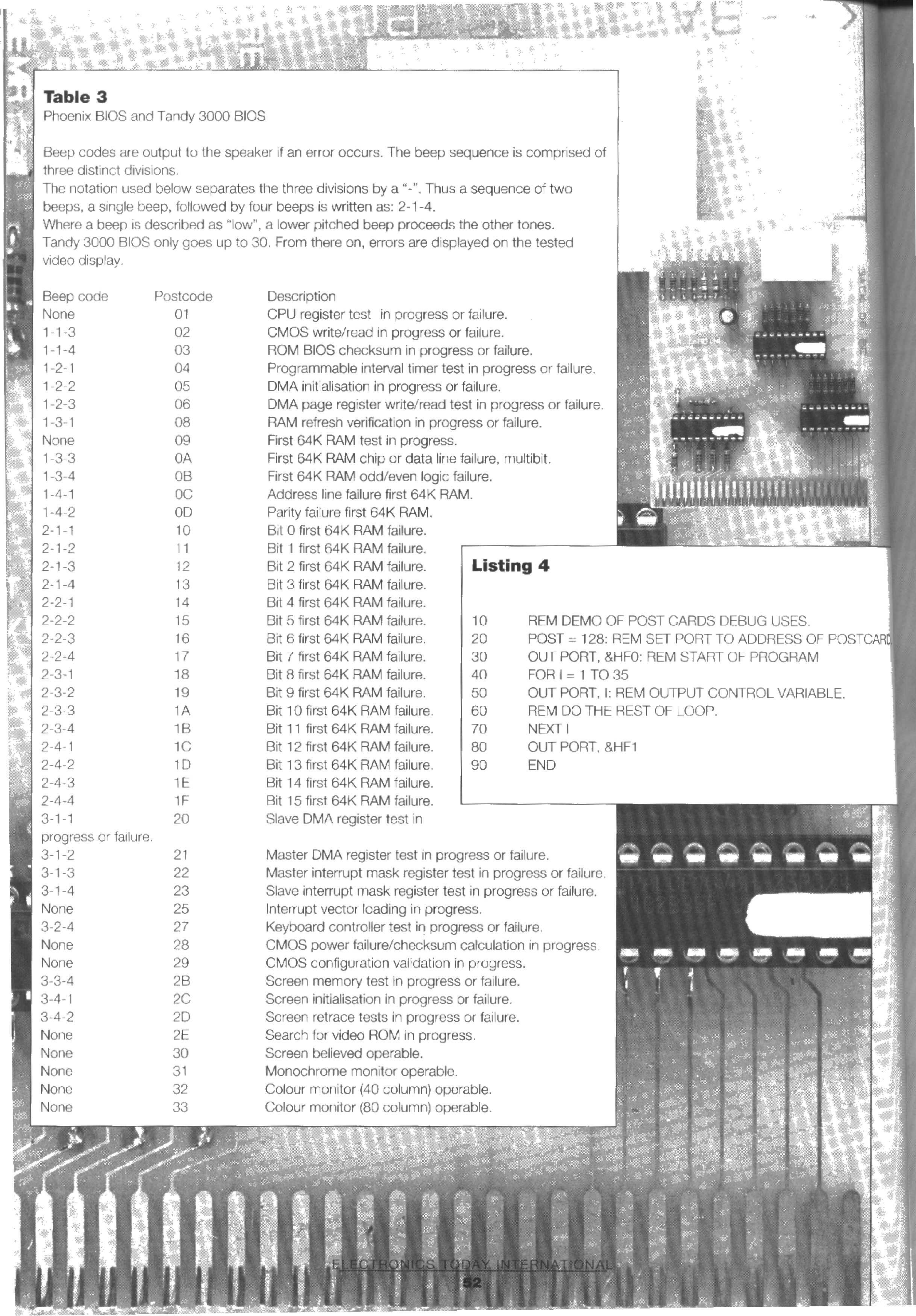


Table 3

Phoenix BIOS and Tandy 3000 BIOS

Beep codes are output to the speaker if an error occurs. The beep sequence is comprised of three distinct divisions.

The notation used below separates the three divisions by a "-". Thus a sequence of two beeps, a single beep, followed by four beeps is written as: 2-1-4.

Where a beep is described as "low", a lower pitched beep proceeds the other tones.

Tandy 3000 BIOS only goes up to 30. From there on, errors are displayed on the tested video display.

Beep code	Postcode	Description
None	01	CPU register test in progress or failure.
1-1-3	02	CMOS write/read in progress or failure.
1-1-4	03	ROM BIOS checksum in progress or failure.
1-2-1	04	Programmable interval timer test in progress or failure.
1-2-2	05	DMA initialisation in progress or failure.
1-2-3	06	DMA page register write/read test in progress or failure.
1-3-1	08	RAM refresh verification in progress or failure.
None	09	First 64K RAM test in progress.
1-3-3	0A	First 64K RAM chip or data line failure, multibit.
1-3-4	0B	First 64K RAM odd/even logic failure.
1-4-1	0C	Address line failure first 64K RAM.
1-4-2	0D	Parity failure first 64K RAM.
2-1-1	10	Bit 0 first 64K RAM failure.
2-1-2	11	Bit 1 first 64K RAM failure.
2-1-3	12	Bit 2 first 64K RAM failure.
2-1-4	13	Bit 3 first 64K RAM failure.
2-2-1	14	Bit 4 first 64K RAM failure.
2-2-2	15	Bit 5 first 64K RAM failure.
2-2-3	16	Bit 6 first 64K RAM failure.
2-2-4	17	Bit 7 first 64K RAM failure.
2-3-1	18	Bit 8 first 64K RAM failure.
2-3-2	19	Bit 9 first 64K RAM failure.
2-3-3	1A	Bit 10 first 64K RAM failure.
2-3-4	1B	Bit 11 first 64K RAM failure.
2-4-1	1C	Bit 12 first 64K RAM failure.
2-4-2	1D	Bit 13 first 64K RAM failure.
2-4-3	1E	Bit 14 first 64K RAM failure.
2-4-4	1F	Bit 15 first 64K RAM failure.
3-1-1	20	Slave DMA register test in progress or failure.
3-1-2	21	Master DMA register test in progress or failure.
3-1-3	22	Master interrupt mask register test in progress or failure.
3-1-4	23	Slave interrupt mask register test in progress or failure.
None	25	Interrupt vector loading in progress.
3-2-4	27	Keyboard controller test in progress or failure.
None	28	CMOS power failure/checksum calculation in progress.
None	29	CMOS configuration validation in progress.
3-3-4	2B	Screen memory test in progress or failure.
3-4-1	2C	Screen initialisation in progress or failure.
3-4-2	2D	Screen retrace tests in progress or failure.
None	2E	Search for video ROM in progress.
None	30	Screen believed operable.
None	31	Monochrome monitor operable.
None	32	Colour monitor (40 column) operable.
None	33	Colour monitor (80 column) operable.

Listing 4

```
10      REM DEMO OF POST CARDS DEBUG USES.
20      POST = 128: REM SET PORT TO ADDRESS OF POSTCARD
30      OUT PORT, &HF0: REM START OF PROGRAM
40      FOR I = 1 TO 35
50      OUT PORT, I: REM OUTPUT CONTROL VARIABLE.
60      REM DO THE REST OF LOOP.
70      NEXT I
80      OUT PORT, &HF1
90      END
```

Table 4

Award BIOS

This is for the following versions:- PC/XT version 3.0 and greater AT version 3.02 and greater Modular BIOS version 3.1

Not all of these POST codes apply to all of the BIOS types listed here. The sequence in which these tests occur is dependent upon the specific BIOS.

01 Processor test 1, processor status verification. Infinite loop if test fails.

02 Determine POST type (normal or manufacturing). Fails if keyboard buffer filled with data.

03 Clear 8042 keyboard controller. Sends TEST_KBRD command (AAh).

04 Reset 8042 keyboard controller. Verify TEST_KBRD.

05 Get 8042 manufacturing status. If manufacturing tests 1 to 5 are continuously repeated.

06 Initialise chips. Disables video, parity, DMA and clears DMA chip, all page registers and CMOS shutdown byte.

07 Processor test 2, verify operation of CPU registers.

08 Initialise CMOS timer. Updates timer cycle normally.

09 EPROM checksum. Checksum must equal zero to pass.

0A Initialise video interface.

0B Test 8254 channel 0.

0C Test 8254 channel 1.

0D Test 8254 channel 2.

0E Test CMOS shutdown byte.

0F Test extended CMOS.

10 Test DMA channel 0.

11 Test DMA channel 1.

12 Test DMA page registers.

13 Test 8741 keyboard controller interface.

14 Test memory refresh toggle circuits.

15 Test first 64K of system memory.

16 Set up interrupt vector table for use by 8259.

17 Set up video I/O operations. Using the optional video BIOS if installed.

18 Test video memory. Bypassed if optional video BIOS installed.

19 Test interrupt controller (8259) mask bits of channel 1.

1A Test interrupt controller (8259) mask bits of channel 2.

1B Test CMOS battery level.

1C Test CMOS checksum.

1D Set configuration from CMOS.

1E Size system memory and compare with CMOS value.

1F Test base memory from 64K to top of memory.

20 Test stuck 8259's interrupt bits.

21 Stuck NMI bits (parity or I/O channel check).

22 Test 8259 interrupt functionality.

23 Test protected mode, 8086 virtual and 8086 page mode.

24 Size extended memory above 1Mb.

25 Test all memory except the first 64K.

26 Test protected mode exceptions.

27 Set-up cache control or shadow RAM.

28 Set-up cache controller or special 8242 keyboard controller.

29 Reserved.

2A Initialise keyboard controller.

2B Initialise floppy drive and controller.

2C Detect and initialise serial ports.

2D Detect and initialise parallel ports.

2E Initialise hard disk drive and controller.

2F Detect and initialise maths coprocessor.

30 Reserved.

31 Detects and initialise optional ROMs present from C800:0 to EFFF:0.

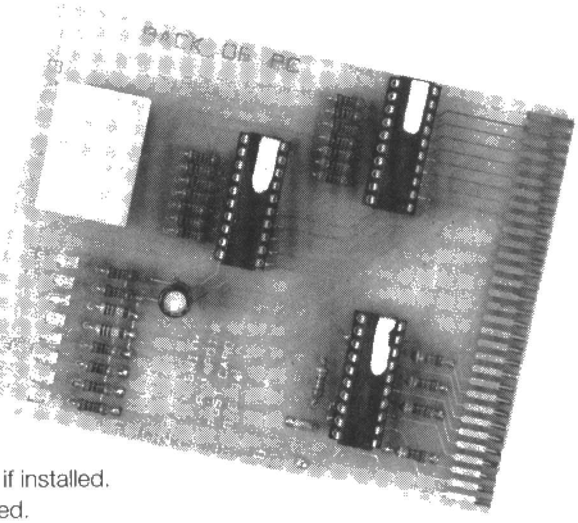
3B Initialise secondary cache with OPTi chipset (486 only).

CA Detect and initialise Micronics cache controller if present.

CC NMI handler shutdown.

EE Unexpected processor exception.

FF Control given to INT 19, boot loader.



Nonfatal system-board errors.

4-2-1	34	Timer tick interrupt test in progress or failure.
4-2-2	35	Shutdown test in progress or failure.
4-2-3	36	Gate A-20 failure.
4-2-4	37	Unexpected interrupt in protected mode.
4-3-1	38	RAM test in progress or address failure > FFFFh.
4-3-3	3A	Interval timer channel 2 test or failure.
4-3-4	3B	Time-of-day clock test in progress or failure.
4-4-1	3C	Serial port test in progress or failure.
4-4-2	3D	Parallel port test in progress or failure.
4-4-3	3E	Maths coprocessor test in progress or failure.
Low 1-1-2	41	System-board select failure.
Low 1-1-3	42	Extended CMOS RAM failure.

Table 5

AMI COLOUR BIOS

01	Processor register test about to start, and NMI to be disabled.
02	NMI is disabled; power on delay starting.
03	Power-on delay complete.
04	Keyboard controller soft reset/power on test.
05	Soft reset/power on determined; going to enable ROM.
06	ROM enabled calculating ROM BIOS checksum and check keyboard buffer clear.
07	ROM BIOS checksum ok, keyboard buffer clear, i ssuing BAT command to keyboard.
08	BAT command issued to keyboard, going to verify BAT command.
09	Keyboard BAT verified, keyboard command byte next.
0A	Keyboard command byte code issued, going to write command byte data.
0B	Keyboard controller command byte written; going to issue pin-23, 24 blocking/unblocking command.
0C	Pin-23, 24 of keyboard controller blocked/unblocked; NOP command issued.
0D	NOP command processing done; CMOS shutdown register test to be done next.
0E	CMOS shutdown register R/W test ok; going to calcu late CMOS checksum.
0F	CMOS checksum calculation done and DIAG byte written; CMOS initialisation to begin.
10	CMOS initialisation done; CMOS status register about to initialise for date and time.
11	CMOS status register initialised, going to disable DMA and interrupt controllers.
12	DMA controller 1 and 2, interrupt controller 1 and 2 disabled; about to disable video display and initialise port-B.
13	Video display is disabled and port-B initialised; chipset init/auto memory detection to begin.
14	Chipset init/auto memory detection over; 8254 timer test about to start.
15	CH-2 timer test halfway; 8254 CH-2 timer test to be complete.
16	CH-2 timer test over; 8254 CH-1 timer test to be complete.
17	CH-1 timer test over; 8254 CH-0 timer test to be complete.
18	CH-0 timer test over; about to start memory refresh.
19	Memory refresh started; memory refresh to be done next.
1A	Memory refresh line is toggling; going to check 15 micro-second on/off time.
1B	Memory refresh period 30 micro-second test complete; base 64K memory test about to start.
20	Base 64K memory test started; address line test to be done.
21	Address line test passed; going to do toggle parity.
22	Toggle parity over; going for sequential data R/W test.
23	Base 64K sequential data R/W test ok; any set-up before interrupt vector initialisation about to start.
24	Set-up required before vector initialisation complete; interrupt vector initialisation about to begin.
25	Interrupt vector initialisation done; going to read I/O port of 8042 for turbo switch.
26	I/O port of 8042 is read; going to initialise global data for turbo switch.
27	Global data initialisation is over; any initialisation after interrupt vector to be done next.
28	Initialisation after interrupt vector is complete; going for monochrome mode setting.
29	Monochrome mode setting is done; going for colour mode setting.
2A	Colour mode setting is done; about to go for toggle parity before optional ROM test.
2B	Toggle parity over; about to give control for any set-up required before optional video ROM check.
2C	Processing before video ROM control is done; about to look for optional video ROM and give control.
2D	Optional video ROM control is done; about to give control to any other processing after video ROM returns control.
2E	Return from processing after the video ROM control; if EGA/VGA not found then do display memory R/W test.
2F	EGA/VGA not found; display memory R/W test about to begin.
30	Display memory R/W test passed; about to look for the retrace checking.
31	Display memory R/W test or retrace checking failed; about to do alternate display memory R/W test.
32	Alternate display memory R/W test passed; about to look for the alternate display retrace checking.
33	Video display checking over; verification of display type with switch setting and actual card to begin.
34	Verification of display adaptor done; display mode to be set next.
35	Display mode set complete; BIOS ROM data area about to be checked.
36	BIOS ROM data area check over; going to set cursor for power on message.
37	Cursor setting for power on message ID complete; going to display the power on message.
38	Power on message display complete; going to read new cursor position.
39	New cursor position read and saved; going to display the reference string.
3A	Reference string display is over; going to display the Hit <Esc> message.
3B	Hit <Esc> message displayed; virtual mode memory test about to start.
40	Preparation for virtual mode test started; going to verify from video memory.
41	Returned after verifying from video memory; going to prepare the descriptor tables.
42	Descriptor tables prepared; going to enter virtual mode for memory test.
43	Entered in virtual mode; going to enable interrupts for diagnostics mode.
44	Interrupts enabled (if diagnostics switch is on); going to initialise data to check memory wrap-around at 0:0.
45	Data initialised; going to check for memory wrap-around at 0:0 and find total system memory size.
46	Memory wrap-around test done; memory size calcula tion over; about to go for writing patterns to test memory.
47	Pattern to be test-written in extended memory; going to write patterns in base 640K memory.
48	Patterns written in base memory; going to determine amount of memory below 1Mb.
49	Amount of memory below 1Mb found and verified; going to determine amount of memory above 1Mb.
4A	Amount of memory above 1Mb found and verified; going for BIOS ROM data area check.
4B	BIOS ROM data area check over; going to check <Esc> and clear memory below 1Mb for soft reset.
4C	Memory below 1Mb cleared (soft reset); going to clear

memory above 1Mb.

4D Memory above 1Mb cleared (soft reset); going to save the memory size.

4E Memory test started (no soft reset); about to display t he first 64K memory test.

4F Memory size display started; will be updated during memory test; going for sequential and random memory test.

50 Memory test below 1Mb complete; going to adjust memory size for relocation and shadow.

51 Memory size adjusted due to relocation and shadow; memory test above 1Mb to follow.

52 Memory test above 1Mb complete; going to prepare to go back to real mode.

53 CPU registers are saved including memory size; going to enter in real mode.

54 Shut down successful, CPU in real mode; Going to restore registers saved during preparation for shut down.

55 Registers restored; going to disable gate A-20 address l ine.

56 A-20 address line disable successful; BIOS ROM data area about to be checked.

57 BIOS ROM data area check halfway; continuing.

58 BIOS ROM data area check over; going to clear Hit <Esc> message.

59 Hit <Esc> message cleared; message displayed; about to start DMA and interrupt controller test.

60 DMA page register test passed; about to verify from video memory.

61 Video memory verification over; about to go for DMA #1 base register test.

62 DMA #1 base register test passed; about to go for DMA #2 register test.

63 DMA #2 base register test passed; about to go for BIOS ROM data area check.

64 BIOS ROM data area check halfway; continuing.

65 BIOS ROM data area check over; about to program DMA unit 1 and 2.

66 DMA unit 1 and 2 programming over; about to initialise 8259 interrupt controller.

67 8259 initialisation over; about to start keyboard test.

80 Keyboard test started, clearing output buffer, checking for stuck key; about to reset keyboard.

81 Keyboard reset error/stuck key found; about to issue keyboard controller interface test command.

82 Keyboard controller interface test over; about to write command byte and initialise circular buffer.

83 Command byte written, global data initialisation done; about to check for lock-key.

84 Lock-key checking over; about to check for memory size mismatch with CMOS.

85 Memory size check done; about to display soft error and check for password or bypass set-up.

86 Password checked; about to do programming before set-up.

87 Programming before set-up complete; going to CMOS set-up program.

88 Returned from CMOS set-up program and screen is cleared; about to do programming after set-up.

89 Programming after set-up complete; going to display power-on screen message.

8A First screen message displayed; about to display message.

8B message displayed; about to do main and video BIOS shadow.

8C Main and video BIOS shadow successful; set-up options programming after CMOS about to start.

8D Set-up options are programmed, mouse check and initialisation to be done next.

8E Mouse check and initialisation complete; going for hard disk, floppy reset.

8F Floppy check returns that floppy is to be initialised, floppy set-up to follow.

90 Floppy set-up is over; test for hard disk presence to be done.

91 Hard disk presence test is over; hard disk set-up to follow.

92 Hard disk set-up complete; about to go for BIOS ROM data area check.

93 BIOS ROM data area check halfway; continuing.

94 BIOS ROM data area check over; going to set base and extended memory size.

95 Memory size adjusted due to mouse and hard disk type 47 support; going to verify display memory.

96 Returned after verifying display memory; going to do initialisation before C800:0 optional ROM control.

97 Any initialisation before C800:0 optional ROM control is over; optional ROM check and control to be done next.

98 Optional ROM control is done; about to give control to do any required processing after optional ROM returns control.

99 Any initialisation required after optional ROM test over; going to set up timer data area and printer base address.

9A Return after setting timer and printer base address; going to set the RS-232 base address.

9B Returned after RS-232 base address; going to do any initialisation before coprocessor test.

9C Required initialisation before coprocessor is over; going to initialise the coprocessor next.

9D Coprocessor initialised; going to do any initialisation after coprocessor test.

9E Initialisation after coprocessor test is complete; goingto check extended keyboard, keyboard ID, and Num Lock.

9F Extended keyboard check is done, ID flag set, Num Lock on/off, keyboard ID command to be issued.

A0 Keyboard ID command issued; keyboard ID flag to be reset.

A1 Keyboard ID flag reset; cache memory test to follow.

A2 Cache memory test over; going to display any soft errors.

A3 Soft error display complete; going to set the keyboard typematic rate.

A4 Keyboard typematic rate set; going to program memory wait states.

A5 Memory wait states programming over; screen to be cleared next.

A6 Screen cleared; going to enable parity and NMI.

A7 NMI and parity enabled; going to do any initialisation required before giving control to optional ROM at E000:0.

A8 Initialisation before E000:0 ROM control over; E000:0 ROM to get control next.

A9 Returned from E000:0 ROM control; going to do any initialisation required after E000:0 optional ROM control.

AA Initialisation after E000:0 optional ROM control is over; going to display the system configuration.

00 System configuration is displayed; going to give control to INT 19 boot loader.

PROJECT

HALL EFFECT WAH-WAH PEDAL

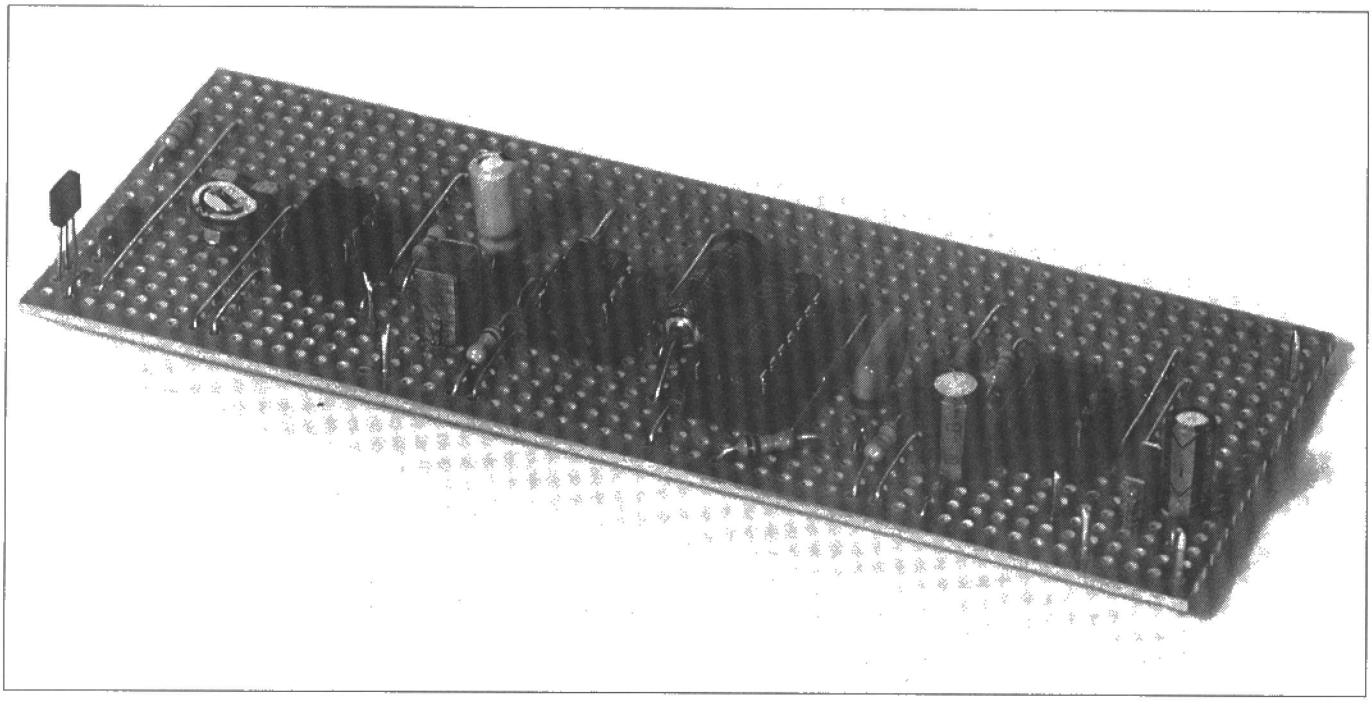
In another of his electronic music projects Robert Penfold shows how to build a simple wah-wah pedal based upon a Hall Effect device

The wah-wah effect is probably one of the earliest of guitar effects units, and it is one which has remained very popular over the years. Producing this effect is relatively easy, and a wah-wah unit is basically just a tuneable bandpass filter. Conventionally, the centre frequency is in the lower bass region with the pedal raised, increasing to a high audio frequency as the pedal is fully depressed. A wah-wah pedal could be used to boost the fundamental frequencies and attenuate the harmonics, but it is normally used the other way round. Sweeping the filter over the lower and middle harmonics boosts them relative to the fundamental frequencies, and produces the familiar "wah-wah" sound.

A conventional wah-wah pedal has a potentiometer that is operated by the pedal. This potentiometer is used as a resistive element in the bandpass filter circuit, and operating the pedal therefore alters the operating frequency of the filter. This arrangement is very simple, but it does have its drawbacks, particularly for the home constructor One problem is that it is difficult to produce a reliable pedal mechanism. It is possible, and plenty of do-it-yourself wah-wah units have been constructed successfully, but many potential constructors are simply not prepared to get embroiled with gears or pulley wheels. Another problem with many wah-wah pedals is the noise produced by the potentiometer each time it is operated This often produces quite noticeable "scratching" sounds.

Hall Effect

This unit is free from both problems due to the use of a voltage controlled bandpass filter, plus a Hall Effect sensor to generate the control voltage. A Hall Effect sensor is a semiconductor which is designed to respond to magnetic fields. It uses the basic arrangement shown in Fig.1(a). A thin slice of silicon has an electrode on opposite sides, and about half way up the slice. The output voltage is taken from across these two electrodes. A current is passed down the slice of silicon, giving a potential gradient which runs from 0 volts at the bottom of the slice to the



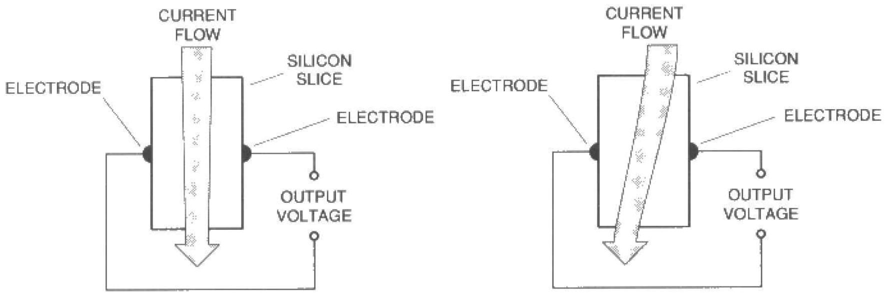


Fig.1. (a) Normal current flow through a Hall Effect device, and (b) distorted current flow caused by a magnetic field

full supply voltage at the top. As the electrodes are both half way up the slice, they both produce half the supply voltage, and the differential output voltage is zero.

If a magnetic field is applied to the device, and it is perpendicular to the current flow, it deflects the current carriers and distorts the current flow (Fig.1(b)). This is rather like a magnetic or electrostatic field deflecting the electron beam of a cathode ray tube. This distortion of the current flow results in a distortion of the potential gradient. This in turn results in the potential at one electrode increasing, and the voltage at the other electrode decreasing. This produces an output voltage across the two electrodes. The stronger the applied magnetic field, the greater the distortion of the current flow, and the greater the output voltage. If the magnetic field is reversed, so is the polarity of the output voltage. However, in order to operate the sensor properly the magnetic field must be applied to a side of the silicon slice that is fitted with an electrode. Applying it to one of the other four sides of the device will have no significant effect on the output voltage.

How It Works

The circuit diagram for the Hall Effect wah-wah pedal appears in Fig.2. IC1 is the linear Hall Effect sensor, and this is a three terminal device which only has the two supply leads and an output terminal. Under standby conditions the output voltage is approximately half the supply voltage. A small bar magnet placed right against the sensor can provide a voltage change of about one volt, which is not quite enough to drive the filter circuit properly. Therefore, IC2 is used as an inverting amplifier to boost the voltage change from the sensor. The closed loop gain of IC2 is approximately 4.5 times, which gives an output voltage swing of roughly 4.5 volts. IC2 also provides level shifting. VR1 is adjusted to give an output voltage of about 0.5 volts or so from IC2 under standby conditions. Bringing the magnet close to IC1 gives a reduction in its output voltage, but an increase in the output voltage from IC2. The output voltage range from IC2 is roughly 0.5 to 5 volts.

A very basic bandpass filter is all that is needed for the present application. IC3 acts as an input buffer stage which ensures that the filter circuit is driven from a suitably low source impedance. The filter uses what is almost a conventional operational amplifier bandpass configuration. It differs from the normal configuration only in that Q7 and IC4 have been included. IC4 is a CMOS 4007UBE complementary pair and inverter. In this circuit though, only one of the N channel enhancement mode MOSFETs is utilised. No connections are made to the other sections of this component. The drain-to-source resistance of the MOSFET is used as a voltage controlled resistance. It is varied via the gate voltage, and because this is an enhancement mode device, it starts to turn on when the gate voltage is about 0.5 volts or so. Increasing the gate voltage above this threshold voltage gives a steady decrease in the drain-to-source resistance.

The components which govern the operating frequency of the filter are Q6, C4, C5, and Q10. The voltage controlled resistance is effectively in parallel with Q6, and any variations in its resistance produce changes in the filter's centre frequency. With a low control voltage to IC4 the filter frequency is about

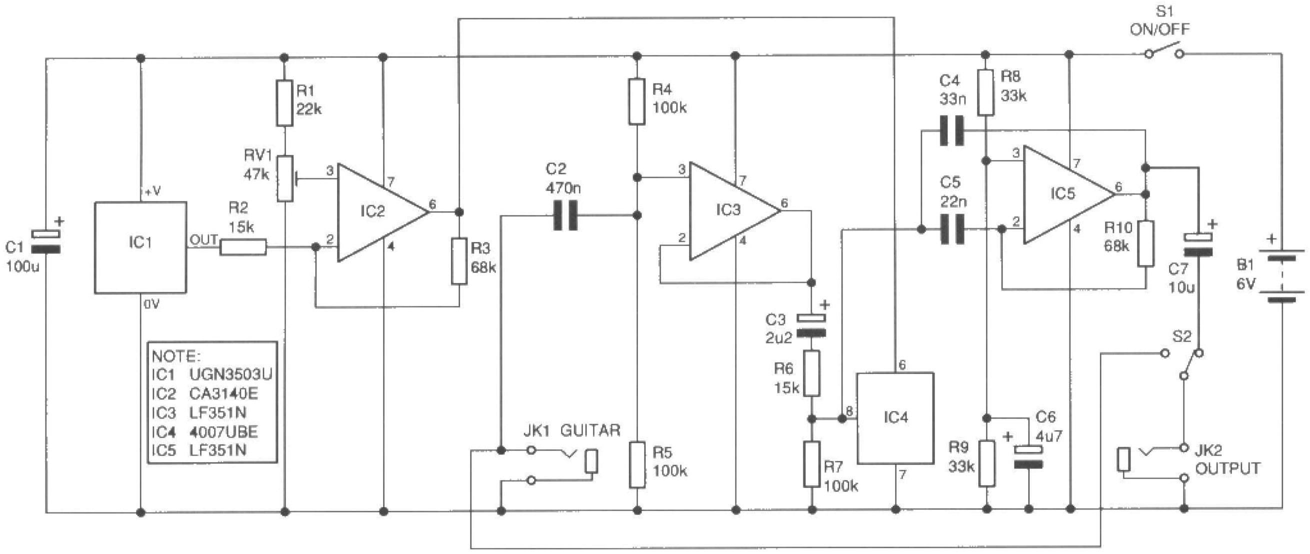


Fig.2. The Hall Effect Wah Wah pedal circuit

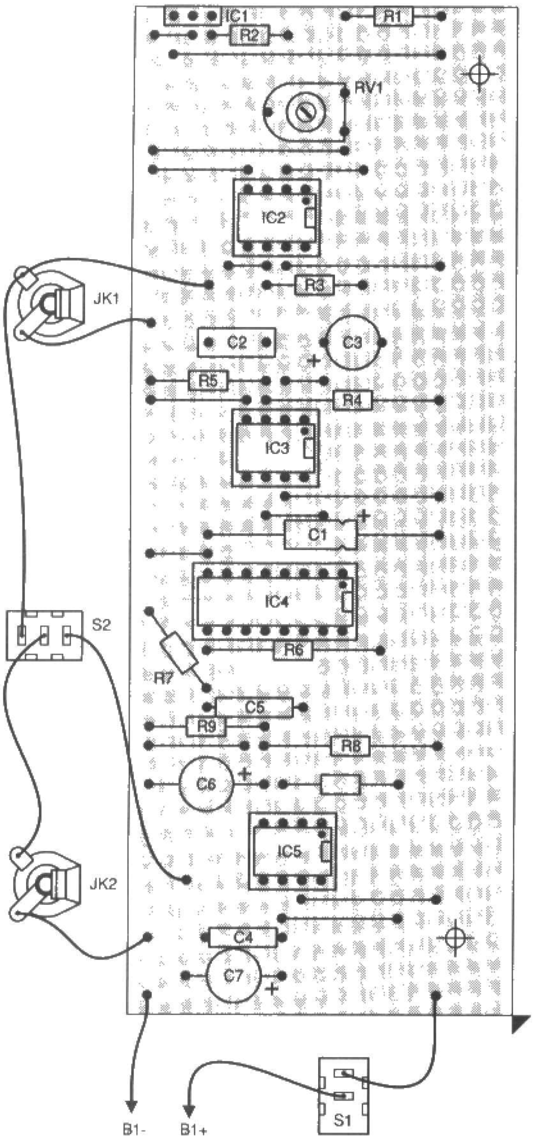


Fig.3. Details of the component layout and wiring

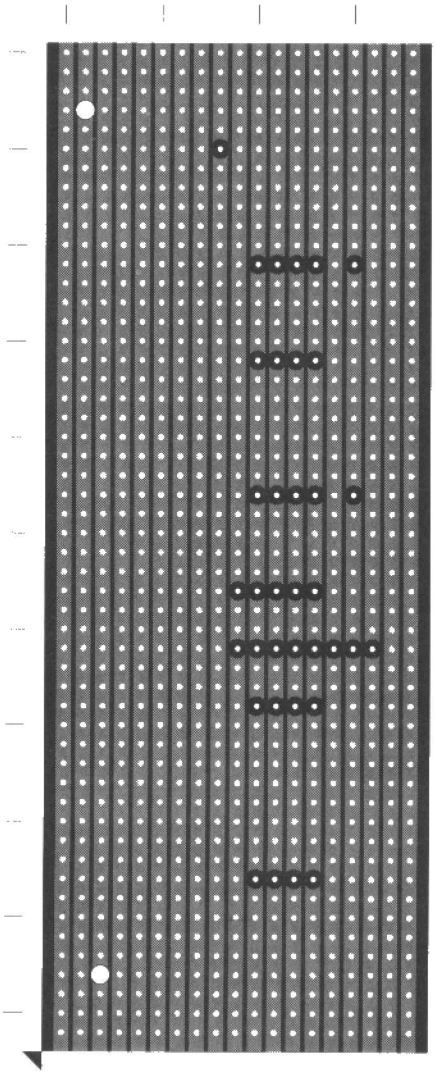


Fig.4. The underside of the component panel

50Hz, with the lower limit being controlled by R4. With the maximum control voltage, the filter frequency is raised to a few kilohertz.

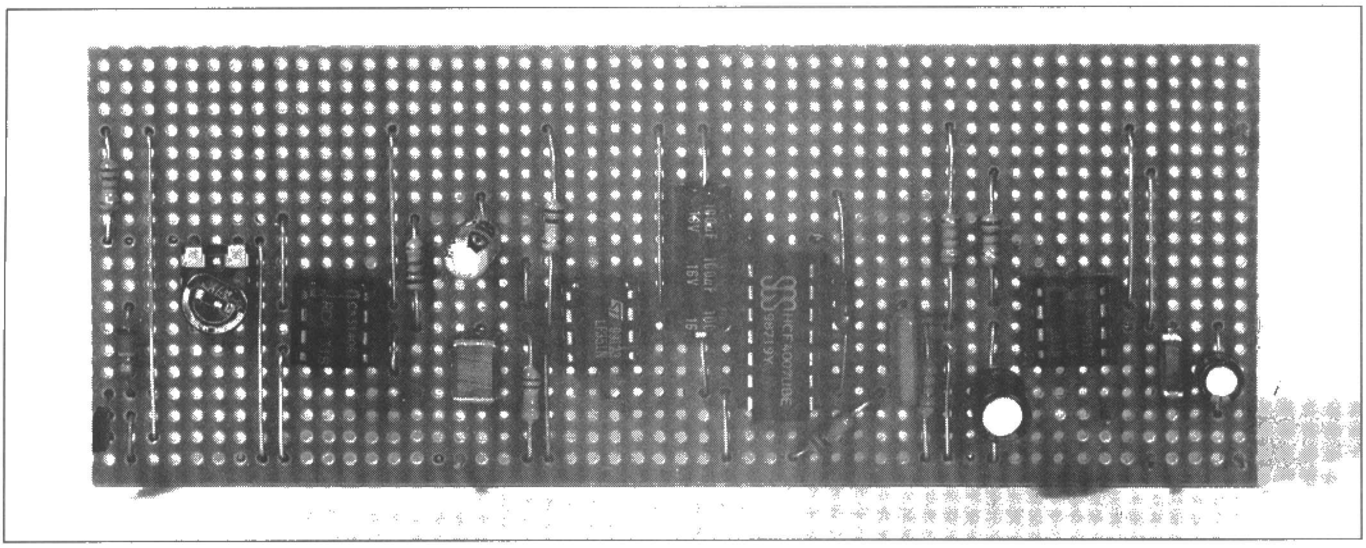
There is a slight drawback to this very simple method of filtering, and this is that the Q of the filter changes significantly when its operating frequency is altered. The Q increases substantially at higher filter frequencies. The circuit values therefore have to be chosen to give a Q that is high enough to give a good effect at low filter frequencies, without giving an excessive Q at high filter frequencies. Although the gain of the filter increases substantially at high frequencies, so too do the losses through Q6. Thus, although the filter's Q changes significantly as the filter frequency is swept, the gain at the centre of the passband remains virtually constant.

S2 is a simple bypass switch that enables the effect to be switched in and out. The current consumption of the circuit is about 15 milliamps, which gives a very long operating life from each set of four HP7 (AA) size batteries.

Construction

Details of the component panel and wiring are provided in Figs.3 and 4. The stripboard panel has 52 holes by 19 copper strips. After cutting the board to size, drill the two 3.3 millimetre diameter mounting holes. These will accept 6BA or metric M3 mounting bolts. Then make the breaks in the copper strips on the underside of the board. Next fit the components and link wires. IC2 and IC4 are both MOS devices which require the standard anti-static handling precautions. In order to fit easily into the layout, C2 and C4 should have 7.5 millimetre (0.3 inch) lead spacing, and C5 should be a 10 millimetre (0.4 inch) type.

Fig.5 shows how to identify the leadouts of the UGN3503U Hall Effect sensor. Using the type number marking to identify the notional front of the device seems to be the only way of doing this. As viewed in Fig.2, the type number is on the right, and the plain side is on the left. If necessary, the sensor can be mounted off-board and wired to the component panel via a piece of three way cable. Note that the sensor will only work if the end of the bar magnet is applied to the front or rear face of the component. The magnet will have little effect if one of its poles is applied to an edge of the sensor, or if the middle of the magnet is applied to one of the faces of the sensor. Trial and error has to be used to determine which end of the magnet



PARTS LIST

Resistors (0.25 watt 5% carbon film)	
Q1	22k
Q2,Q6	15k
Q3,Q10	68k
Q4,Q5,Q7	100k
Q8,Q9	33k
VR1	47k min hor preset
Capacitors	
C1	100u 10V axial elect
C2	470n polyester
C3	2u2 50V radial elect
C4	33n polyester
C5	22n polyester
C6	4u7 50V radial elect
C7	10u 25V radial elect
Semiconductors	
IC1	UGN3503U
IC2	CA3140E
IC3,IC5	LF351N
IC4	4007UBE
Miscellaneous	
JK1,JK2	Standard 6.25mm jack socket
S1	SPST min toggle
S2	SPDT heavy-duty push-button (see text)
B1	6 volt (4 x HP7 size cells)

Case, 0.1 inch matrix stripboard having 52 holes by 19 strips, battery connector (PP3 type), plastic battery holder, Maplin "large" magnet, materials for pedal, 8 pin DIL IC holder (3 off), 14 pin DIL IC holder, wire, solder, etc.

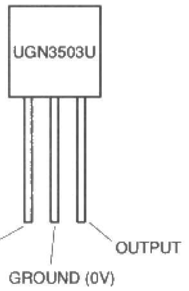


Fig.5. Identifying the UGN3503U leadout wires

operates the unit in the correct fashion. The unit will probably work properly using practically any small bar magnet, but I have only tested it using the Maplin "large" type.

A pedal style project needs to be housed in tough case, such as a diecast aluminium box or a good quality folded aluminium case. There should be no difficulty in producing a simple pedal mechanism based on a large hinge and a small sheet of 16 s.w.g. aluminium. The magnet only needs to move through about 12 millimetres in order to take the unit through its full sweep range. S1 must be a heavy-duty push-button switch mounted next to the pedal if you wish to be able to switch the effect in and out while playing. It should be a switch of the successive operation type (i.e. pressing it once switches the effect out, pressing it again switches the effect back in again, and so on).

The four HP7 (AA) size batteries are fitted into a plastic battery holder of the appropriate type. The connections to the holder are made by way of an ordinary PP3 battery connector.

Adjustment And Use

When initially testing and adjusting the unit the best signal source is some form of noise generator. A suitable signal can be obtained from the earphone socket of an F.M. radio tuned between stations. Start with VR1 adjusted fully clockwise, and the magnet withdrawn from the sensor. The audio output from the unit should be a fairly ferocious sounding, high-pitched noise signal. If VR1 is slowly adjusted in a counter-clockwise direction, the pitch of the noise should soon start to fall. Adjust VR1 just far enough to bring the noise down to its minimum pitch. Introducing and withdrawing the magnet should then sweep the filter over its full frequency range. The unit is then ready for use.

A little practice is needed in order to get the best from a wah-wah pedal. The frequency range covered by the filter needs to be matched to the notes played, so the high the pitch of the notes, the higher the sweep range that should be used. You really need to listen carefully to the effect being produced, and modify your pedalling to maximise the effect. Remember that "practice makes perfect".

MIDI EXPLAINED

This is the sixth and final part of Robert Penfold's tutorial series which has sought to unravel some of the mysteries of MIDI-based electronic musical instruments

In recent years there has been endless speculation about a Mark II version of MIDI. Although there could be real benefits in having a new and improved version of MIDI, it has become so well established in its original form that it seems unlikely that we will be using a new version of MIDI in the near future. On the other hand, over the years, MIDI has undergone a few minor changes and has had some substantial additions. The original MIDI 1.0 specification remains at the heart of the current MIDI specification, but we now have what could reasonably be called a Mark II version of MIDI.

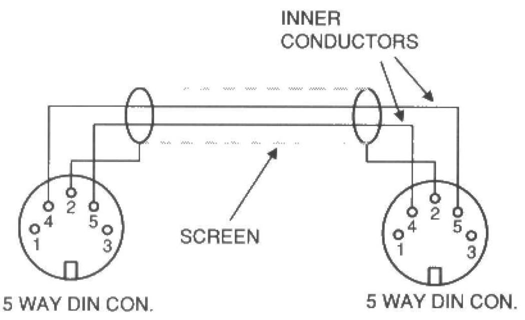


Fig.1. MIDI cable interconnections

General MIDI

Many of the changes and updates to MIDI have been covered in the previous articles in this series. One obvious and major exception is General MIDI. This is more correctly called General MIDI System, Level 1, but it is better known as just "GM". This has become something of a buzzword in the world of electronic music, but relatively few people seem to fully understand just what GM is really all about. The first point that has to be made is that GM is not something that has relevance to all MIDI users. Indeed, to many it is of no consequence at all. The main MIDI specification is intended to ensure that items of equipment from various manufacturers can be used together as a proper system. For many of us, and possibly for the majority of MIDI users, this is all that it needs to do.

One problem with the original MIDI specification is that it does not lay down any minimum requirements for a MIDI system. Whether your MIDI system consists of a couple of old monophonic synthesisers, or the latest computer plus sophisticated software and twenty instruments, it is a perfectly valid MIDI system. This is fine in a situation where musicians are using their own

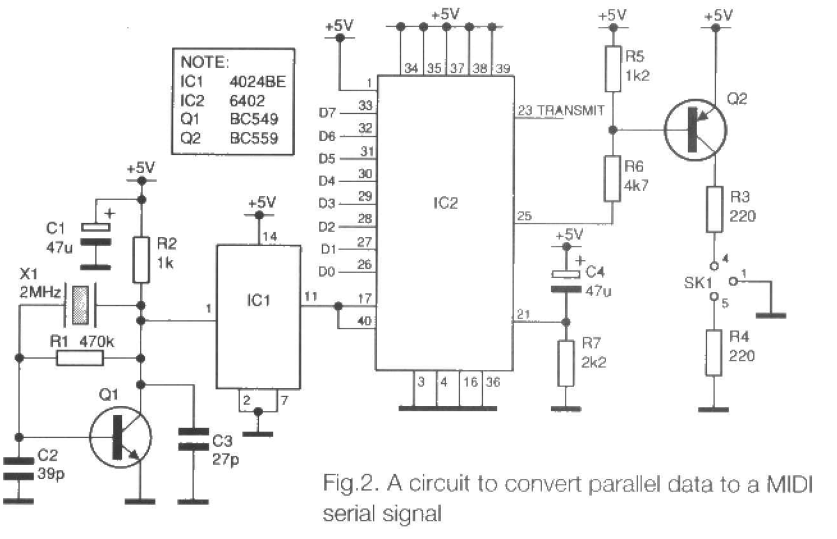


Fig.2. A circuit to convert parallel data to a MIDI serial signal

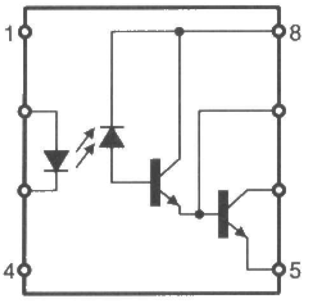


Fig.3. Pinout details of the 6N139 and internal circuit

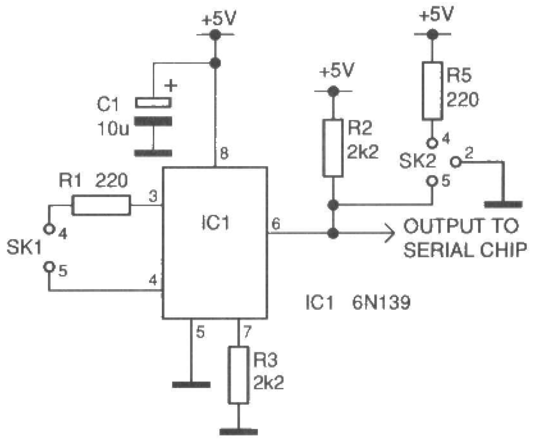


Fig.4. A MAID output stage using the 6N139

systems to produce music in their own ways. Each musician can set up his or her own MIDI system to play his or her own sequences. The problem only really arises when musician "A" wishes to play a sequence produced by musician "B". This is usually quite possible, and with the introduction of standard MIDI files it is not even necessary for musician "A" to have the same sequencer as musician "B". However, the system used by musician "A" must have the wherewithal to play all the notes, and it must be set up to provide the right sound on each channel.

GM is basically just a specification for a standard MIDI system which can be used to play standard MIDI files in a predictable way. To a large extent GM is about getting the right sound on the right channel. This is not done by having a standard sound assigned to each channel, since the 16 available MIDI channels would then give a limit of just 16 different sounds. Instead, there are standard sound assignments for the 128 program numbers, giving a much more useful repertoire of sounds. The general idea is for each sequence to start with a program change message for each channel that is used in the sequence. The purpose of these messages is to set each channel to the appropriate sound. Of course, further program change messages can be used mid-sequence in order to vary the sound assigned to each channel, as and when necessary. Due to the standard sound assignments of GM, the correct sound will always be produced on each channel.

GM is not perfect, and one obvious problem is that 128 programs are not sufficient to accommodate a full range of sounds. This is not a major problem for most users, since the usual string sounds, wind sounds, etc. are catered for, as are some of the more unusual instruments and a range of sound effects. Those who operate in specialist areas of music, and composers of avant garde music might feel that GM is a bit limiting though.

Another problem is that each sound is only loosely defined. The flute sound of one instrument might sound significantly different to the flute sound of another instrument. Obviously there are some variations in the sounds of "real" instruments of a given type, but there are greater variations with synthesised sounds. The intention is that each sound should eventually be more rigidly defined, with the GM specification laying down such things as the attack and decay times of the envelope shape. However, even without such rigid standardisation, GM at least ensures that each track of a sequence will be played by an appropriate type of sound, and that the reproduced piece will be largely in accordance with the composer's intentions.

Beat Channel

It is now common practice to have one channel of a MIDI system devoted to percussive sounds. The alternative is to use mode 4 and to have a different percussive sound on each channel, but this method does not make efficient use of the channels. If you need, say, six percussion sounds, only ten channels would be left for the other tracks. It is much better to have all the percussion sounds on one channel, with a different sound being assigned to each note value. This provides up to 128 different sounds, but only uses up one MIDI channel.

Under GM it is MIDI channel 10 that is assigned to the percussive sounds, and there is a standard set of sound assignments for note values from 35 to 81 (decimal). Of course, this channel is only used for simple percussive sounds that do not have variable pitch. Percussive sounds such as woodblock and steel drums are used on the other channels in the normal way.

The GM sound assignments are arranged in 16 groups of eight, as detailed in Table 1. Table 2 gives details of the assignments within a few groups, and Table 3 provides some examples of the sound assignments on the percussion channel (MIDI channel number 10). Note that only program numbers from 35 to 81 have so far been assigned sounds.

Other Factors

GM is not just about sound assignments, and the GM standard gives minimum requirements for the system. One of these is that the system should have 24 fully dynamically allocated voices for both melodic and percussive sounds, or 16 dynamically allocated voices for melody and eight for percussion. This enables quite complex sequences to be handled, but with modern technology this sort of thing can be handled by a single instrument. All 16 MIDI channels must be supported, and the system must be able to use a different timbre (sound) on each channel. Middle C must be at the correct note value of 60 (decimal). This may seem to be an example of stating the obvious, but apparently some MIDI instruments are an octave out! All voices, including those on the percussion channel, must respond to velocity information. Channel pressure and pitch bending should be implemented on all channels. The system must respond to certain control changes, such as master volume, pan, and sustain.

It must be stressed that GM is not intended to be a definition of a standard system that must be utilized by all MIDI users. It is only for those who need a standard system so that they can swap standard MIDI files in the knowledge that the played-back sequences will always sound more or less as intended. I suppose that GM is primarily for those who wish to buy or sell music in the form of disks containing standard MIDI files. Of course, sequences intended for a GM system can be played back properly on any system that is set up correctly and has the wherewithal, regardless of whether or not it is an official GM system (complete with the GM logo).

TABLE 1

General MIDI System, Level 1 Sound Assignment Groups (not ch. 10)

Program No.	Group
1 - 8	Piano
9 - 16	Chromatic Percussion
17 - 24	Organ
25 - 32	Guitar
33 - 40	Bass
41 - 48	Strings
49 - 56	Ensemble
57 - 64	Brass
65 - 72	Reed
73 - 80	Pipe
81 - 88	Synth Lead
89 - 96	Synth Pad
97 - 104	Synth Effects
105 - 112	Ethnic
113 - 120	Percussive
121 - 128	Sound Effects

Hardware

Proper hardware standardisation is undoubtedly one of the main reasons for MIDI's success. To a large extent it was a lack of proper hardware standards that resulted in previous music interfaces generating a lot of disillusionment. All MIDI devices produce signals that are fully compatible with all other MIDI units. In order to wire-up a MIDI system you merely need some standard MIDI cables. The MIDI standard permits professional quality 3 way XLR connectors to be used for IN, OUT, and THRU sockets, but these are little used in practice. The normal MIDI connector is the 5 way 180 degree DIN type. A MIDI connecting lead consists of two 5-way 180 degree DIN plugs having a "straight" connection between pins 4 and 5, and pin 2 of each plug connected to the screen. This method of interconnection is shown in Fig.1.

Making up your own MIDI leads is quite easy, and any reasonably good quality twin screened cable is suitable. MIDI operates at the relatively high baud rate of 31250 baud, which limits the maximum cable length that can be used. The maximum recommended length for MIDI cables is 15 metres (about 50 feet), but this should be more than sufficient for most users. Many ready-made MIDI cables seem to include "straight" connections between pins 1 and 3. This is not in accordance with the MIDI specification, but as these pins of the sockets should not be connected to anything, the extra connections should not be of any consequence. Note that 5-way DIN leads for audio use normally have some form of cross-coupling between the plugs. Leads of this type are unusable in a MIDI system.

Computers And MIDI

Probably most MIDI systems are based on a computer running a sequencer program. A few computers have a built-in MIDI interface, but the only common examples are the Atari STs. Although the MIDI ports of musical instruments seem to rigidly adhere to the MIDI specification, the situation is rather different for computers. Unfortunately, computer MIDI ports are about as well standardised as everything else in the computer world! In the case of the Atari STs, pins 1 and 3 of the OUT port are used to provide the THRU facility. A special lead is therefore

TABLE 2

Program No.	Instrument
1	Grand Piano
2	Bright Acoustic Piano
3	Electric Grand Piano
4	Honky-Tonk Piano
5/6	Electric Pianos
7	Harpsichord
8	Clavi
25	Acoustic Guitar (nylon strings)
26	Acoustic Guitar (steel strings)
27	Electric Guitar (clean)
29	Electric Guitar (muted)
30	Overdriven Guitar
31	Distortion Guitar
32	Guitar Harmonics
41	Violin
42	Viola
43	Cell
44	Contrabass
45	Tremolo Strings
46	Pizzicato Strings
47	Orchestral Harp
48	Timpani
57	Trumpet
58	Trombone
59	Tuba
60	Muted Trumpet
61	French Horn
62	Brass Section
63/64	Synth Brass
65	Soprano Sax
66	Alto Sax
67	Tenor Sax
68	Baritone Sax
69	Oboe
70	English Horn
71	Bassoon
72	Clarinet
73	Piccolo
74	Flute
75	Recorder
76	Pan Flute
77	Blown Bottle
78	Shakuhachi
79	Whistle
80	Ocarina
113	Tinkle Bell
114	Agogo
115	Steel Drums
116	Woodblock
117	Taiko Drum
118	Melodic Tom
119	Synth Drum
120	Reverse Cymbal

TABLE 3

some examples of the sound assignments on the percussion channel (MIDI channel number 10).

Program No.	Percussion Sound
35	Acoustic Bass Drum
36	Bass Drum 1
37	Side Stick
38	Acoustic Snare
39	Hand Clap
44	Pedal Hi-Hat
47	Low Mid Tom
48	High Mid Tom
50	High Tom
54	Tambourine
60	High Bongo
61	Low Bongo
69	Cabasa
70	Maracas
71	Short Whistle
72	Long Whistle
75	Claves
76	High Woodblock
77	Low Woodblock
81	Open Triangle

needed in order to utilize the THRU facility. In other cases, the wrong type of connector is used, and a non-standard lead is again needed in order to use any of the ports.

Most computers require an add-on MIDI interface. The standard MIDI interfaces for the Amiga and Macintosh computers are serial port add-on units. Bearing in mind that MIDI uses what is essentially a normal RS232C style asynchronous serial signal, this is an obvious approach to the problem. Most computer serial ports have a baud rate that is controlled by the system clock and a divide by "N" counter. Setting the correct baud rate is just a matter of writing the correct value to the counter. Some simple hardware is all that is needed to make the conversion from RS232C to MIDI signal levels, and vice versa.

MIDI interfacing to the PCs seems to be a fairly involved subject. Some MIDI users have found a low cost solution in the form of modified serial ports, but this method seems to have gone out of fashion, probably due to a lack of compatibility with commercial software. The nearest thing to a standard PC MIDI interface is the Roland MPU-401, which is available these days in more compact form as the MPU-1PC. This is rather more than just a basic input/output port, and it actually includes its own microprocessor. This gives a boost in performance, as the interface can undertake some of the processing that would otherwise have to be undertaken by the main processor.

Most of the alternative PC MIDI interfaces offer MPU- 1PC compatibility, but note that some do not offer full compatibility. The built-in processing power of the MPU-1PC is sometimes absent, and a MIDI interface card of this type will obviously not function properly with software that tries to use the missing processor. Virtually all PC MIDI software is MPU- 1PC compatible, and most major pieces of software also offer compatibility with several other interface cards. Provided you stick with mainstream hardware and software there is little risk of compatibility problems arising, but it is clearly a good idea to make sure before actually buying anything.

DIY Hardware

Any normal serial interface chip should be suitable for use in a MIDI device. The word format is one start bit, eight data bits, one stop bit, and no parity. Although the baud rate is quite high at 31250 baud, it is within the capabilities of any normal serial interface chip. For a non-micro based circuit the standard 6402 UART (or an equivalent) is probably the best choice. The circuit of Fig.2 uses a 6402 to take in parallel data and convert it to a MIDI output signal. The correct word format is obtained by taking pins 34 to 39 to the appropriate logic levels. A clock signal at 16 times the baud rate is required, which works out at 500kHz. This is provided by a 2MHz crystal clock oscillator and a divide by four action through IC1. The 6402 has separate transmitter and receiver clock inputs at pins 40 and 17 respectively. C4 and R7 provide a reset pulse to the UART at switch-on.

MIDI does not operate at normal logic levels, but instead uses a 5 milliamp current loop. TR2 is used as a simple common emitter switch that provides a 5 milliamp "on" current to the opto-isolator at the MIDI input it is used to drive. The "Transmit" input should normally be high, and is pulsed low in order to transmit the current byte on D0 to D7. Bear in mind that bytes cannot be sent at a rate of much more than about three bytes per millisecond.

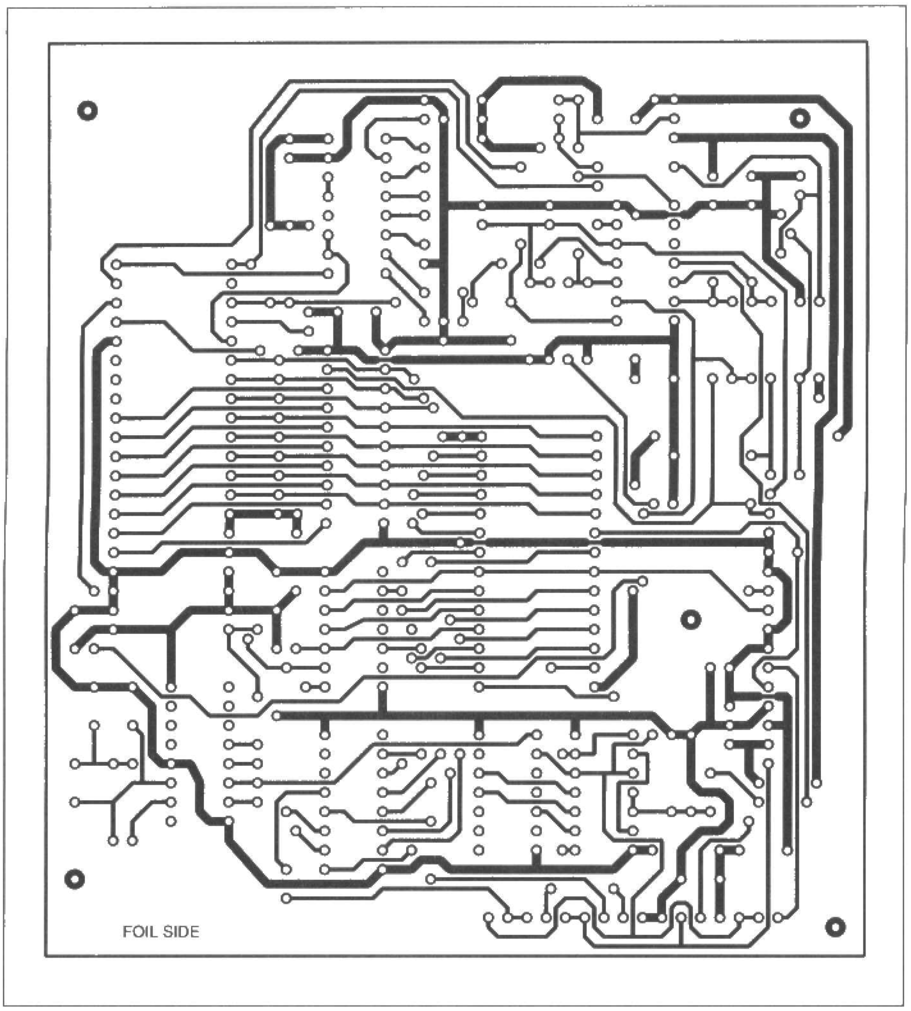
Do not bother trying to use a "bog standard" opto- isolator at a MIDI input. Devices such as the TIL111 are about a thousand times too slow for this application. The MIDI standard calls for maximum rise and fall times of 2us, which necessitates the use of a high quality opto-isolator. The devices mentioned in the MIDI specification are the PC-900 and the 6N138, but neither of these are readily available in the U.K. The 6N139 is very similar to the 6N138, is readily available and, in practice, it works well in MIDI input stages.

Pinout details for the 6N139, together with its internal circuit, are provided in Fig.4. It has the usual infra- red LED on the input side. On the output side there is a photodiode driving an emitter follower transistor. This stage in turn drives a common emitter switching transistor. The minimum efficiency of the device is 400%, and it can typically transfer data up to 300k baud (i.e. about ten times the MIDI baud rate). Fig.4 shows the circuit diagram for a MIDI input stage based on the 6N139. SK1 is the MIDI IN socket, and this is connected to the LED at the input of IC1 via protection resistor Q1. Q2 and Q3 are respectively the load resistors for the emitter follower and common emitter stages. SK2 is the THRU socket, and this is driven from the output transistor of IC1 via current limiting resistors R4 and R5. If this input stage is used with the UART circuit of Fig.2, its output is connected to pin 20 of the UART. The last decoded byte of data is available on pins 5 (D7) to 12 (D0).

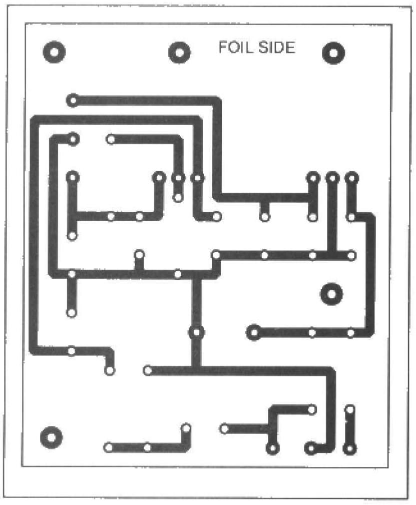
Finally

This concludes our look at what makes MIDI "tick". For most users it is only necessary to have a fairly basic understanding of the MIDI messages, and the general way in which MIDI functions. It is then easy to exploit MIDI's potential, without wasting time trying to implement a facility that is not available, or overlooking a valuable feature that is readily accessible. The situation is rather different for anyone producing MIDI hardware or software. A detailed understanding of the various messages is then required, including a knowledge of the "fine print". Either way, these articles should have told you what you need to know. Ultimately though, the only way to get to grips with MIDI is to start using it in earnest, and to learn from your experience.

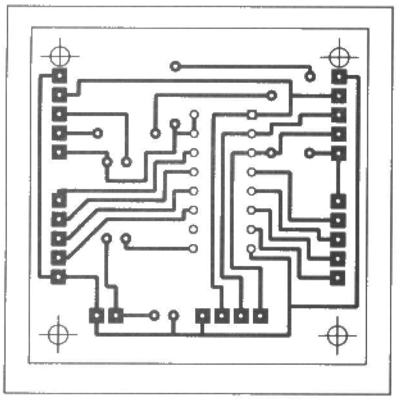
Foils for this issue



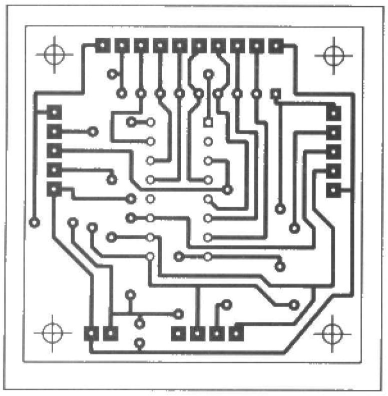
Eprom programmer



Eprom programmer & Eprom emulator power supply



Parallel output



Parallel input